PETER LAURIE

INFORMATICA PARA TODOS

BIBLIOTECA CIENTIFICA SALVAT

PETER LAURIE

INFORMATICA PARA TODOS

SALVAT

Versión española de la obra original inglesa *The joy of computers* publicada por Hutchinson & Co. (Publishers) Ltd. de Londres
Asesoramiento informático: Joan Oliver
Revisión técnica: Mónica Díaz

Edición digital: Sargont (2018)

ISBN 84-345-8246-5 Obra completa
ISBN 84-345-8405-0
Depósito Legal: NA-136-1986
Publicada por Salvat Editores, S.A. - Mallorca, 41-49 – Barcelona
Impresa por Gráficas Estella. Estella (Navarra)
Printed in Spain

Índice de capítulos

Introducción

Este libro pretende mostrar algunos aspectos de ese mundo nuevo y fascinante que, hasta hace muy poco, ha sido privativo de unos pocos miles de profesionales altamente remunerados.

En la actualidad, cuando los pequeños computadores se han extendido por todo el mundo gracias a su abaratamiento, millones de personas empiezan a preocuparse por su ignorancia en este campo, pero mucho más por la de sus hijos, para quienes puede llegar a suponer una desventaja aún peor que el analfabetismo. No cabe la menor duda de que hoy en día, en la economía de muchos países, el sector de la informática es uno de los pocos que muestran señales de expansión. Jóvenes especialistas en informática recién salidos de la universidad, obtienen sustanciales sueldos como profesionales en empresas de *hardware* y de *software*. Personas con mucha menos formación, quizá con sólo unos pocos meses de experiencia en el estudio de BASIC por su cuenta, reciben ofertas de empresas para trabajar con microcomputadores.

Existe un mito, que ha sido cuidadosamente alentado por las grandes compañías del sector, según el cual hay algo de mágico en torno a los computadores y las personas que los utilizan. Se ha extendido la leyenda de que los computadores son “cerebros electrónicos” y las personas que los programan una especie de superhombres. La verdad es que los computadores carecen por completo de inteligencia y las personas que los programan son seres humanos normales. Cualquiera que pueda contar con los dedos de 0 a 7 y obtener un 8 puede aprender a ser programador. La cosa en sí no es difícil, sólo hay que conocer los trucos.×

Constituye un grave error creer que los computadores pueden pensar como las personas. No pueden. De hecho no poseen más inteligencia propia que la que pueda tener una cortadora de césped. Sin embargo, permiten a quienes los manejan almacenar información. Si encontramos la manera de realizar una determinada tarea o resolver un problema concreto, y lo podemos escribir en forma de programa, el computador aplicará entonces nuestro pensamiento a esa tarea o problema tantas veces como deseemos. En este sentido, los computadores y los programas tienen cierta vida, ya que perpetúan el pensamiento de quienes los han con-

feccionado. A menudo se oye decir a los profesionales del sector: «¿Cómo sabe esta subrutina que debe hacer tal y tal cosa?», hablando de una subrutina como si se tratara de una persona. Cuando en realidad, hablando en propiedad, deberían decir: «¿Cómo transmitió el programador la información a esa subrutina para ordenarle hacer tal y tal cosa?»

La revolución informática promete realizar profundos cambios en nuestra forma de vida, pero estos cambios no serán más complicados que otros muchos que han sido fácilmente asimilados. En períodos recientes de la historia, la humanidad ha vivido las revoluciones de la imprenta, la producción industrial, el ferrocarril, la electricidad, el telégrafo, el teléfono, la aviación, la radio y la televisión. La informatización supone simplemente un paso más en la ininterrumpida marcha de la humanidad hacia la consecución del dominio sobre su entorno. Se inventaron las máquinas para aligerar y potenciar el trabajo de nuestra mente.

A la larga, no cabe la menor duda, la informática originará cambios que no podían siquiera imaginarse al principio del proceso.

Aunque la informatización se inició durante la segunda Guerra Mundial y, por tanto, cuenta con muy pocos años de existencia, ya ha dado lugar a una cultura propia, rica y compleja. Es imposible comprender los actuales microcomputadores sin tener alguna idea de los procesos anteriores, ya que incorporan supuestos e ideas que han ido acumulándose gradualmente a lo largo de los años.

Pero aunque la historia es importante, el ritmo de cambio es tan rápido que cualquiera que tenga una idea brillante tiene una excelente oportunidad de destacar en el campo de la industria. El cambio se está produciendo simultáneamente en dos frentes. El hardware de los computadores se abarata y se hace cada día más potente. Esto significa que el trabajo que hace unos años sólo podían afrontar equipos de especialistas con máquinas de gran tamaño, hoy puede realizarse de forma rutinaria en miles de oficinas. Por otra parte, los computadores se popularizan y se convierten en un instrumento cotidiano de trabajo en todo el mundo. Estas máquinas ya no son utilizadas exclusivamente por una casta sacerdotal de elevados ingresos, que habla un lenguaje sólo comprensible para los iniciados, sino también por personas corrientes, más interesadas en realizar un trabajo concreto que en los computadores en sí. Este hecho está produciendo en estas máquinas cambios análogos a los que experimentaron los automóviles bajo la influencia de la producción en masa.

En un principio, los automóviles eran un juguete en manos de los entusiastas. Se podía viajar desde Londres a Pekín, pero había que estar preparado para reconstruir el vehículo entero varias veces durante el viaje. Tan pronto como se empezaron a fabricar automóviles con vistas a un mercado de consumo mayoritario, fue preciso un cambio. Los nuevos vehículos tuvieron que ser dignos de confianza, estandarizados, confortables. El nuevo tipo de propietario de automóvil, lejos de estar dispuesto a reajustar los pistones cada treinta kilómetros, se enfurecía si la puerta rechinaba o si fallaba el encendedor de cigarrillos. Lo mismo está ocurriendo en el campo de la informática. Hasta hace muy poco, la típica persona que tenía un computador era un fanático, capaz de reconstruir su máquina dos veces en una noche sobre la mesa de la cocina. Ahora, hay miles de personas que esperan poder conectar sus microcomputadores para efectuar sus cálculos, procesar un texto o entretenerse con algún juego y a las que desconcierta por completo la simple idea de tener en sus manos un soldador.

Quizás el lector piense que el mundo de la informática es un mundo totalmente ordenado, regido por una lógica esterilizada. De ningún modo. De hecho, este mundo es sorprendentemente similar al mundo de la alta costura. También los computadores se ven afectados, por manías y modas, excéntricos, fanáticos, charlatanes y lunáticos, así como por un gran número de personas trabajadoras, interesadas y razonables, fascinadas por encontrarse en la vanguardia del progreso humano, que en todo momento hacen lo que les parece más adecuado para ayudar a que el proceso de desarrollo continúe.

Desde otro punto de vista, el mundo de la informática se asemeja al del salvaje Oeste americano del siglo pasado. El territorio es tan vasto, la riqueza tan enorme, que nadie tiene tiempo de sentarse y meditar. La industria está ávida de manos e ideas nuevas. Pide más rendimiento que calificación. En el Oeste, si alguien disparaba con puntería y tenía cara de persona honesta, era nombrado *sheriff*. En el mundo de la informática quien sabe hacer un trabajo, obtiene un puesto, con independencia de donde haya aprendido a hacerlo y sin que importen los títulos que posea.

Son varias las razones que atraen a la gente hacia este nuevo sector. Una de ellas, sin lugar a dudas, es el hecho de que ofrece nuevos empleos en un momento en que éstos andan escasos. En segundo lugar, ofrece un campo abierto a toda clase de talentos: el mundo entero está sufriendo un

proceso de informatización y la industria necesita gente que sepa de cualquier cosa. En tercer lugar, las inversiones prometen un buen rendimiento. El desarrollo de la comercialización masiva abre posibilidades de ganar fortunas al estilo de Hollywood. Los dos jóvenes fundadores de Apple Computer, que tuvieron que vender una furgoneta y una calculadora para financiar su primera máquina, cinco años después eran millonarios.

Ya sea por ganar dinero o por ansias de aventura, siempre ha habido en este sector personas dispuestas a desarrollar sus ideas sin importarles adonde llevasen, y el resultado ha sido una sorprendente variedad de máquinas diferentes y lenguajes diversos y de técnicas distintas para realizar todo tipo de trabajos. En consecuencia, el tema es tan amplio, que este libro podría reescribirse cuatro o cinco veces sin duplicar la mayor parte del material. En él, no puedo hacer otra cosa que señalarles esa multitud de perspectivas fascinantes; espero que mis lectores piensen que vale la pena examinarlas detenidamente.

PETER LAURIE

1. El computador

EL MICROCOMPUTADOR

Es un computador pequeño, por supuesto, que no difiere en lo fundamental de los grandes IBM que precisan de enormes dependencias para instalar sus innumerables armarios. La definición clásica de un computador pone de relieve que es una máquina cuya función primordial es procesar datos, aunque esta definición por sí misma quizá no sirva de gran ayuda. Esperamos que la lectura de este libro ayude a una mejor comprensión de cómo son y para qué sirven los computadores. Por lo que no es preciso dar ahora una definición académica. En cualquier caso, este libro es simplemente un intento de hacer llegar al lector una parte del fascinante y divertido mundo de estas máquinas; no es en modo alguno un libro de texto.

Quizá sea interesante dar ahora alguna idea de estas máquinas. Los computadores se dividen en tres grandes clases: *main-frames,* minicomputadores y microcomputadores. La diferencia entre ellos, que al principio era puramente técnica, es hoy en día mucho más una cuestión de precio y de marketing.

Un *main-frame* es una máquina de las mayores y más caras; precisa de un equipo de profesionales para su manejo y de un local acondicionado, que puede costar tanto como el propio computador.

Los minicomputadores son el fruto de los primeros esfuerzos en la tecnología informática en el sentido de lograr el abaratamiento y la miniaturización de los computadores. Aparecieron hace unos diez años para proporcionar a los usuarios —que eran generalmente departamentos universitarios o empresas de cierta envergadura— máquinas que no precisasen de un equipo de profesionales dedicados exclusivamente a su servicio ni de locales especialmente acondicionados. Son bastante más baratos que los main-frames, pero, aún así, están fuera del alcance de la mayoría.

La característica esencial de los microcomputadores es que, virtualmente, todo el mundo puede comprarlos y utilizarlos. De momento, éstos se dividen a su vez en dos clases: personales, los más baratos y menos

potentes, pero que tienen un papel esencial en la expansión de la computarización; y los equipos, que a menudo cumplen muchas de las funciones de los minicomputadores o de los main-frames.

Las tres edades del hardware

1. El computador personal, doméstico o de iniciación. Se sirve del televisor para la visualización y una cassette para la grabación y muchos disponen de una impresora rudimentaria. Casi siempre tienen 64 K de memoria, o menos.
2. La máquina para pequeñas empresas (IBM llama al suyo, astutamente, “Computador personal”, *Personal Computer).* Puede ser de 8 o 16 bits, dispone de discos, una impresora matricial o una de margarita y un grueso manual de instrucciones. Tiene normalmente 128 K de memoria y es capaz de ejecutar una amplia gama de programas estándar.
3. El computador “auténtico” en el que cada terminal puede tener un usuario distinto. Cada uno de ellos podrá disponer de accesorios tales como un digitalizador, un dispositivo trazador de gráficos *(plotter)* y un ratón. Utiliza discos duros y el computador propiamente dicho puede ser un multiprocesador de 8 o 16 bits, un minicomputador o un main-frame clásico.

Esta clasificación se hace día a día más confusa debido a la rápida evolución de los *chips* —circuitos integrados—, que se fabrican cada vez más potentes. Los primeros microcomputadores de 8 bits no eran demasiado potentes, pero las nuevas máquinas de 16 bits son a menudo tan potentes como los minicomputadores; mientras que los microcomputadores de la última generación, tales como el Hewlett-Packard de 32 bits, poseen una potencia de cálculo similar a la de los main-frames más pequeños.

De hecho, todo esto no es demasiado relevante, porque lo realmente interesante e importante de la informática no reside en el *hardware* —la propia máquina—, sino en el *software* —los programas.

La forma más práctica de imaginarse un microcomputador es compararlo a una máquina de escribir eléctrica. Al pulsar una tecla, aparece la letra correspondiente en una pantalla a través de un proceso sorprendentemente complicado, del que hablaremos más adelante. De ese modo pueden escribirse varias letras que formen palabras y frases e incluso un libro entero; simultáneamente, puede verse en la pantalla y grabarse todo

el texto en disco o cinta magnética. Posteriormente, puede reproducirse e incluso modificarse si se desea. Puede sustituirse automáticamente "Cárter" por "Reagan" en el caso de que se haya producido un cambio en la presidencia de Estados Unidos. Por ejemplo, puede obtenerse un programa que elabore el índice del libro para indicar a qué página pertenece cada palabra. No es demasiado difícil hacer todo esto, pero resulta pesado y es mucho mejor tener una máquina que lo haga que hacerlo uno mismo.

Puede que tenga usted que hacer con frecuencia largas y aburridas sumas o llevar la contabilidad de su negocio o de su departamento. ¿Qué ocurre si los salarios aumentan un 5%, el volumen de ventas se incrementa en un 30%, el coste de los materiales disminuye en un 6%, la tasa de interés disminuye en un 1,1% y se abre un nuevo mercado en Arabia Saudí?

El microcomputador puede calcularlo todo. Si usted está diseñando un puente, tendrá que asurarse de que cada viga sea lo bastante fuerte para aguantar el peso que le corresponda más el peso del tránsito sobre la estructura. Si uno de los elementos resulta demasiado débil, habrá que sustituirlo por otro más fuerte, lo que modificará el peso soportado por los demás y obligará a calcularlo de nuevo. Todo esto podría hacerle perder mucho tiempo, a no ser que decida que lo haga una máquina.

Supongamos que usted es aparejador, responsable de calcular las cantidades necesarias de material para la construcción de un determinado edificio. ¿Cuántos ladrillos se necesitan para construir un muro de 13 metros de altura por 50 metros de longitud, con aberturas para dieciséis ventanas y cuatro puertas? ¿Qué cantidad de mortero se necesita para colocar los ladrillos? ¿Qué cantidad de hormigón para los cimientos? Un microcomputador puede calcularlo todo.

Quizá le guste jugar a "Invasores del espacio", pero no le apetece ir a un local público de máquinas de juego. En este caso también un microcomputador tiene algo que ofrecerle.

Los microcomputadores son buenos para hacer los trabajos aburridos, permitiendo a sus propietarios hacer otras cosas más creativas. No es fácil hacerlos trabajar, y esto, en sí, ya tiene un cierto atractivo. Pero la verdad es que no hay nada de fascinante en un computador haciendo simplemente su trabajo.

ENTRANDO EN MATERIA

Un computador ejecuta un programa que procesa determinados datos —la entrada—, obteniendo unos resultados —la salida. El programa puede hacer algo tan simple como verificar qué letra del teclado se ha pulsado y representarla en la pantalla. La tecla pulsada es la entrada, el dato; la letra en la pantalla, la salida.

Para el computador es indiferente que se pulse la tecla correspondiente a una letra, a un número, a un signo de puntuación o una tecla que no imprima nada. Todas ellas están codificadas, en el conjunto de caracteres de ASCII *(American Standard Code for Information Interchange,* véase p. 292), como números que van del 0 al 127. La letra ‘A’ es el 65, al ‘espacio’ le corresponde el 32, el número ‘3’ es el 51, el signo V es el 43, y así sucesivamente.

Sin embargo, el computador sólo entiende una cosa: la presencia o ausencia de corriente eléctrica, lo cual interpreta como ‘sí o ‘no’, ‘conectado’ o ‘desconectado’, o ‘1’ o ‘0’. De ese modo convierte los números del sistema decimal en sus equivalentes en el sistema binario. Usted no necesita saberlo hacer, le basta con recordar que mientras las cifras o dígitos decimales van del 0 al 9 en cada columna, las cifras o dígitos binarios van del 0 al 1. Por tanto, dos en el sistema binario se escribe 10, cuatro se escribe 100, ocho se escribe 1.000 y dieciséis 10.000. En informática, a los dígitos binarios se les llama ‘bits’ o, abreviadamente, *b.*

Al ejecutar el supersimple programa de escritura en la pantalla, el usuario golpea la ‘A’, Cuyo código ASCII es 65, en binario 01000001. Este grupo de ocho bits es lo que el ordenador lee, identificándolo suidamente en la ROM o memoria sólo de lectura *(read-only memory,* véanse pp. 31-32) y representa finalmente ‘A’ en la pantalla.

Si se hubiesen golpeado las teclas correspondientes a ‘3 + 4’, el teclado hubiera enviado los códigos 51, 43, 52, en sus formas binarias, y el computador hubiese mostrado los tres caracteres ‘3’, ‘+’, ‘4’ en la pantalla. Evidentemente, usted sabe que ‘3 + 4’ es igual a ‘7; pero, si desea que lo haga la máquina, necesitará un programa bastante complicado que examinará la línea que usted ha escrito, identificando el código correspondiente al signo ‘+’ (y también a ‘–‘, ‘•’, ‘/’, de restar, multiplicar y dividir) y los correspondientes a los sumandos ‘3’ y ‘4’. El lenguaje BASIC incluye un programa de ese tipo, y es este programa el que hace

que el computador «sepa» que puede sumar '3' y '4', pero no 'A' y 'B' —a menos, por supuesto, que 'A' y 'B' se utilicen para representar números en un programa algebraico.

Si se mira el cuadro de la página 292, se verá que el sistema ASCII no se ajusta muy bien a los múltiplos de 10. El 16, base del sistema hexadecimal, resulta un múltiplo más adecuado. En hexadecimal se cuenta 1, 2, 3, ..., 9, A, B, C, D, E, F. El número 16 es importante en informática, así como también el 8. Ocho bits forman un byte y el byte se ha convertido en la unidad estándar de información útil. (La abreviatura de byte es *B*.) La capacidad de la memoria interna de los computadores y la de los discos se mide en bytes. La razón por la que el byte es una unidad útil es que proporciona suficiente espacio para almacenar todos los caracteres del teclado.

Puesto que un byte consta de 8 bits, puede tomar 2^8 valores distintos. Dos, multiplicado por sí mismo 8 veces, es 256; por lo tanto, un byte puede utilizarse para codificar 256 cosas diferentes. Dado que el sistema ASCII va de 0 a 127, utiliza sólo 7 bits, es decir, la mitad de las 256 posibilidades. Si esto resulta complicado, recuérdese que cuando trabajamos en el sistema binario, al añadir un bit (o sea un dígito en binario) por la izquierda, multiplicamos por 2 el número de valores que podemos representar; al igual que, en el sistema decimal, al añadir otra cifra por la izquierda, multiplicamos por 10 el número de valores representables. Con un bit podemos representar el 0 o el 1; con dos bits, del 0 al 3; con tres bits, del 0 al 7...; con siete bits, del 0 al 127 y con ocho bits del 0 al 255 (o sea, 256 posibilidades).

Los restantes 128 códigos disponibles pueden utilizarse de dos formas distintas. Al transmitir la codificación binaria ASCII a través de una línea telefónica, siempre existe la posibilidad de error. En previsión de ello, al octavo bit se le asigna un 1 o un 0 según sea par o impar el número total de unos que haya en el resto del byte. En el otro extremo, la terminal correspondiente del computador comprueba si es correcto. Si algún byte ha resultado alterado, el octavo bit estará equivocado, y la terminal receptora pedirá entonces que le sea repetida la transmisión. Esto es lo que se llama «control de paridad» y al octavo bit se le conoce como «bit de paridad».

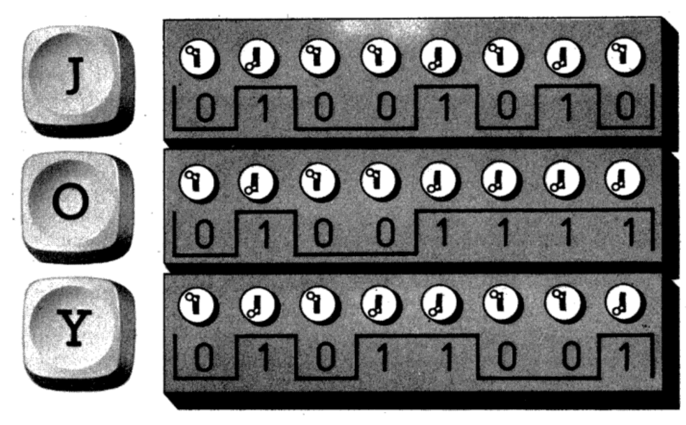


Fig. 1. La palabra “JOY” almacenada en el código ASCII. Los 8 bits de cada uno de los 3 bytes pueden representarse mediante posiciones de los interruptores, o lo que es lo mismo como ‘1 ’ y ‘0’.

La otra forma de utilizar el octavo bit o los caracteres superiores al 127 ASCII, es dentro del propio computador (donde la transmisión de errores es improbable), para proporcionar a los usuarios un conjunto de caracteres «gráficos» del mismo tamaño que las letras; éstos pueden ser utilizados por los más decididos para pintar monigotes en la pantalla.

LA PLACA DEL COMPUTADOR

Cuando se destapa la carcasa de cualquier computador (desde los IBM más grandes hasta los Sinclair más pequeños), lo que se ve dentro es una maraña de conexiones y piezas. Las formas oblongas negras corresponden a lo que comúnmente se conoce como circuitos integrados, abreviadamente CI o *chips;* aunque el chip propiamente dicho es una minúscula ficha cuadrada de medio centímetro, guardada dentro de una funda de plástico negro herméticamente cerrada.

A estos objetos oblongos se les llama circuitos integrados, porque combinan en un único objeto lo que antes se obtenía asociando multitud de transistores, condensadores y otros componentes de los circuitos.

Las fundas de plástico negro son mayores que el chip propiamente dicho, haciéndolo así más manejable y facilitando su conexión eléctrica

en la placa de circuitos del computador. Esta conexión se efectúa mediante las patillas de conexión (que sobresalen alineadas a cada lado del chip a modo de patitas), insertándolas en agujeros previamente marcados en la placa, para luego doblarlas y soldarlas.

Existen cientos de tipos diferentes de chips que realizan otras tantas funciones distintas. Hay chips que desempeñan las funciones lógicas OR, NOT, AND, XOR (véanse pp. 25-26). Otros son capaces de seleccionar y guardar un único bit de una transmisión de datos; otros, de recordar grandes masas de datos, de transformar transmisiones en paralelo en transmisiones en serie (véanse pp. 28-31), de realizar operaciones aritméticas e incluso de convertir un idioma escrito en una lengua hablada (véase p. 178). Los distintos chips se identifican por el número impreso en la parte superior.

Un chip por sí solo no es de gran utilidad. Tiene que ser alimentado por energía eléctrica, con señales procedentes de otros chips o dispositivos externos, y sus salidas deben pasar al dispositivo siguiente. De todo esto se encargan las pistas metálicas impresas sobre la fibra de vidrio de que está hecho el panel de circuitos del computador. Hay otro conjunto de pistas impresas en la parte posterior del panel. Algunas máquinas son tan complicadas que requieren tres o cuatro niveles de pistas para la interconexión de sus chips. La parte más importante del trabajo de diseño de un computador consiste en escoger los chips necesarios para llevar a cabo las funciones específicas de la máquina y en disponer una placa de circuitos en la que puedan montarse.

Esta tarea se simplifica gracias a la existencia de paquetes de software que funcionan en los computadores actualmente existentes, que realizan la mayor parte del trabajo de interconectar los chips y preparar la placa de circuitos.

Fabricar el computador consiste simplemente en imprimir la placa de circuitos, insertar correctamente los chips en los correspondientes agujeros, soldar las conexiones y someter a prueba el resultado. Todo esto puede hacerse casi automáticamente, lo que convierte la fabricación de un computador en algo más parecido a la impresión de un libro que a la construcción de una casa.

Dado un diseño correcto, resulta muy barato fabricar placas de circuitos de esta manera y el proceso continúa abaratándose día a día. Actualmente, los únicos componentes caros son la carcasa y la unidad de sumi-

nistro de energía. Pronto llegará el día en que la mayor parte del coste de un computador residirá en la caja que lo contenga.

MEMORIA Y PROCESADOR

Desde el punto de vista del usuario, quizá lo más importante de un computador sea su memoria. (Su memoria interna, no la memoria externa del disco o de la cinta magnética.) A igualdad de todos los demás factores, cuanta más memoria tenga el computador, mejor; ya que podrá ejecutar programas más amplios y, además, sobre mayor cantidad de datos. Todos los microcomputadores de 8 bits (categoría en la que actualmente están incluidos la mayoría de los que existen en el mercado), tienen un máximo de 2^{16} = 65.536 ubicaciones de memoria que pueden ser direccionadas a la vez. Por esta razón, los procesadores de 8 bits usan en realidad 16 bits para su direccionamiento de memoria.

Puede ser interesante reflexionar sobre el hecho de que si el contenido de cada espacio de memoria de un microcomputador se escribiese en una ficha y se colocasen todas estas fichas una al lado de la otra, cubrirían una distancia de unos 9 km; y que, si se hiciese la misma operación con la memoria de las últimas máquinas de 16 bits, la distancia cubierta sería de más de 1.600 km. Imagínense lo que sería tener que recorrer arriba y abajo semejante fila, cogiendo una ficha en un determinado punto, leyéndola, corriendo hacia un punto lejano del horizonte para leer la ficha a la que hacía referencia la primera, romperla para seleccionar dos más y realizar todas las sumas indicadas sobre la marcha. Esta analogía no es del todo razonable, ya que ningún computador hace nunca nada que usted no pueda hacer con lápiz y papel. Se trata únicamente de que lo hace mucho más deprisa y con más exactitud, hasta el punto de permitir, de hecho, un salto cuantitativo en las posibilidades de trabajo. Muchas operaciones adquieren interés y utilidad si se realizan a la escala y la velocidad requeridas.

El corazón de todo computador es el procesador. Al igual que el motor en un coche, constituye una parte esencial de la máquina, pero se necesitan muchas más piezas y elementos para que el conjunto funcione. Existen muchos tipos distintos de procesadores, pero todos funcionan básicamente de la misma manera. La diferencia más importante entre ellos estriba en la “longitud de la palabra” que pueden tratar. Esta “pala-

bra" es estrictamente un término de los diseñadores de chips y no tiene demasiado que ver, por ejemplo, con las palabras de esta página. La "palabra" es la unidad básica de datos que acepta la máquina. En estos momentos, la mayoría de los microcomputadores utilizan una palabra de 8 bits. Es decir, que contemplan el mundo en trozos de 8 bits (o 1 byte).

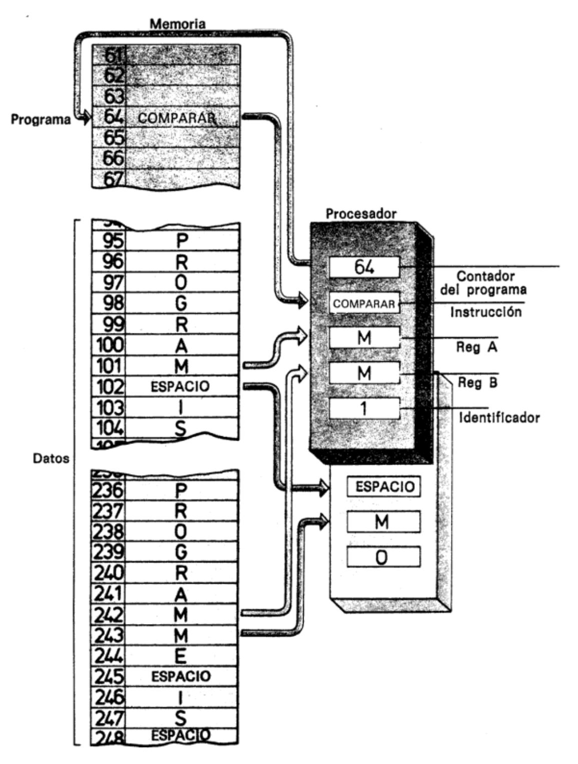


Fig. 2. Memoria de un procesador buscando la palabra "programma" para convertirla en "programa". Las instrucciones se cargan en el extremo superior de la memoria (números bajos). La instrucción en curso, "comparar", se carga en el registro de instrucción, y su número en la memoria en el contador del programa. Carga las letras correspondientes de los trozos de memoria que contienen las palabras "programa" y "programma" en los registros A y B. Si las letras son las mismas el resultado de la instrucción de comparación es un '1' en el identificador (en primer plano) y un '0' si son distintas (en segundo plano). El siguiente paso consiste en mirar al identificador para saber qué hacer luego.

Simplificado al máximo, un procesador no es sino un chip con tres posiciones de memoria (sólo 8 transistores en fila). Los expertos las llaman “registros”. Una de las posiciones contiene una “instrucción”; las otras dos contienen bytes de datos. Según sea la instrucción, el procesador puede sumar los dos bytes, sustraer uno del otro, o compararlos para ver si son iguales. Y esto es todo lo que puede hacer. Por supuesto, incluso esto no es fácil. Para llevar a cabo cada uno de estos procesos, los 8 bits en el registro hacen entrar en juego a todo un conjunto de otros transistores, que conectan las dos posiciones de datos, para producir el efecto deseado.

Repasando la breve lista de acciones aparentemente inútiles presentada más arriba, se podría preguntar qué utilidad pueden tener.

La respuesta es que si usted puede sumar y restar (la comparación es simplemente una resta en la que se pretende obtener un 0 como resultado), también puede multiplicar y dividir. Y, si puede sumar, restar, multiplicar y dividir, puede hacer cálculos tales como raíces cuadradas y logaritmos. Y, si puede hacer esto, puede hacer cualquier cálculo matemático.

De hecho, la mayor parte del tiempo, el procesador hace cosas mucho más triviales, tales como buscar la letra “m” que sobra en el texto, donde se ha escrito “programma” en vez de “programa”, de manera que el paquete de tratamiento de textos pueda cambiarlo.

Para cambiar el error del texto, el paquete de tratamiento de textos compara el código para “mma” con los códigos de diversas letras en el texto, hasta que se encuentra una correspondencia; entonces inserta los códigos para “m”. Puesto que, según vimos en la página 13, las letras están representadas por números, todo lo anterior se reduce a la comparación de dos números.

Los procesadores que se utilizan en realidad son mucho más complejos que el simple procesador de tres registros que hemos descrito. Pero funcionan esencialmente de la misma manera, así como también lo hacen los nuevos de 16 bits, los antiguos de 32 y 64 bits de grandes mainframes y lo harán los nuevos procesadores de 16, 32 y 64 bits, que formarán parte de los computadores que utilizaremos en el futuro. Por suerte, los usuarios de los microcomputadores no necesitamos saber cómo se consigue que los procesadores hagan todas las cosas útiles que pueden hacer.

De las cuestiones operacionales y de funcionamiento de los computadores ya se han ocupado las personas que escribieron los lenguajes que utilizamos; y si no escribimos programas en BASIC o en Pascal (lo que es muy probable) sino que simplemente ejecutamos paquetes de programas para el tratamiento de textos, efectuar cálculos o jugar a "Invasores del espacio", nos encontramos todavía alejados, ya que, casi con toda suridad, las personas que escribieron estos paquetes utilizaron un lenguaje de alto nivel y es improbable que ellos mismos supieran cómo lograr que un simple procesador haga todas estas cosas (véase LENGUAJE MÁQUINA Y ESTRUCTURA DE DATOS, p. 104).

El procesador por sí solo no es más útil que la memoria aislada. Ambos deben trabajar de manera conjunta.

Los microcomputadores actuales utilizan sus memorias para dos propósitos completamente distintos: para almacenar el programa y para almacenar datos sobre los que el programa debe trabajar. Supongamos que estoy buscando la 'm' extra de 'programma'. Lo que ocurre es que el código ASCII de "m" —109 o 01101101— se carga en el registro A del procesador. Los sucesivos códigos ASCII que representan este texto van pasando a otro registro, por ejemplo el B. Se ordena entonces al procesador que compare A con B. Si son iguales, aparece una indicación. El procesador obtiene sus instrucciones de otra área de memoria, cuyo contenido alimenta el registro de instrucciones.

La ventaja del sistema que hemos mencionado es que permite al programador mezclar en la misma memoria los datos y el programa en la proporción que precise.

CHIPS

Tal vez ha llegado el momento de investigar con más atención los minúsculos chips que pueden hacer todo lo que hemos descrito en las páginas 17-20.

Como veremos en las páginas 235-237, con un número suficiente de transistores puede reproducirse cualquier proceso lógico; por tanto, resulta posible imitar a cualquier máquina. Además, de este modo se podría llevar a cabo cualquier procedimiento que pueda ser especificado lógicamente, por complicado y enrevesado que sea. Por ejemplo, en principio no existe ninguna razón por la que no podamos, con ayuda de transisto-

res, construir máquinas capaces de extraer minerales del suelo y autorreproducirse: una especie de virus electrónico que podría ser enviado al espacio y realizar copias de sí mismo cada vez que hallase una playa arenosa. Dotando a un artefacto semejante de rudimentarios poderes de observación y autoprotección, dispondríamos de una máquina capaz de colonizar el Universo. Por supuesto, más tarde nos lamentaríamos de haber construido una máquina semejante.

Pero dejemos las fantasías para concentrarnos en los transistores. Como puede verse, los transistores están formados por líneas de un material de aspecto esponjoso; son los conductores, que llevan electricidad de un sitio a otro. Allí donde se juntan, estos conductores originan un transistor, que no es sino un interruptor electrónico.

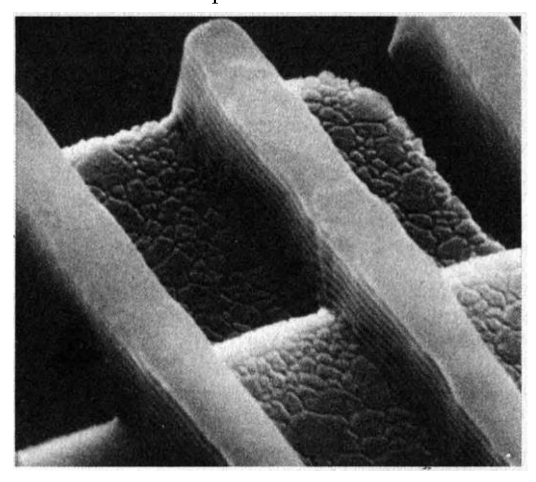


Fig. 3. El espesor de estas estrías en su porción más fina es de alrededor de dos millonésimas de metro de grosor. El chip ampliado a la escala de esta fotografía tendría unos 6,5 km de ancho.

Para tener una idea de la escala a la que nos moveremos, es importante saber que las estrías de la figura 3 tienen alrededor de 2 millonésimas de metro (2 micras) de grosor. Ello significa que utilizando la tecnología que se emplea para fabricar los chips de los microcomputadores estandarizados de 8 bits podría obtenerse un plano de las calles de la ciudad de Londres que mostrase todas las plazoletas y callejones sobre una ficha

cuadrada de medio centímetro. O bien se podría escribir apretadamente en ella un texto de 15.000 palabras. Cuatro chips colocados uno junto a otro formando un cuadrado de 2,5 cm de lado, podrían contener todo el texto de este libro, o todo un periódico. La tecnología que se emplea en la fabricación de los procesadores de 16 y 32 bits permitiría almacenar un plano de Londres, Nueva York, París o Moscú en uno de estos chips. Es asombroso que sea posible representar cosas tan grandes (como para que uno pueda desorientarse y perderse en ellas tan fácilmente) sobre algo tan pequeño que pueda esfumarse confundido con la borra del bolsillo de una chaqueta.

Tecnologías que todavía se hallan en período experimental, y que se utilizarán para la fabricación de los computadores de los próximos cinco años, posibilitarán poner un mapa detallado de todo el sur de Inglaterra o de California desde San Francisco a Los Ángeles en un chip de medio centímetro.

Actualmente, estos objetos minúsculos ya figuran entre las máquinas más complicadas construidas por el hombre; que puedan ser impresos por unos pocos dólares, es realmente extraordinario.

TRANSISTORES Y PUERTAS

Transistores

Un transistor no hace ni más ni menos de lo que hace un relé clásico. Un relé consiste en una barra de hierro con una bobina enrollada (un electroimán). Cuando existe un voltaje entre los terminales de entrada, circula una corriente por la bobina, el imán empuja hacia abajo la pestaña de hierro y ésta gira sobre su eje. La pestaña empuja entonces los dos contactos elásticos, uniéndolos de manera que permitan el paso de una corriente entre los terminales de salida. En consecuencia, una tensión en la entrada produce el paso de una corriente en la salida.

Los transistores hacen exactamente lo mismo, pero son mucho más pequeños y, más que construirse, se imprimen, por lo que su fabricación resulta mucho más barata.

Para fabricar un transistor se funde silicio —que puede encontrarse en grandes cantidades en cualquier playa— en un horno, obteniéndose un

solo cristal del tamaño de un panecillo, el cual, a continuación, se corta en finas y brillantes plaquitas redondas.

El silicio puede ser sometido a tres operaciones eléctricas: se le puede oxidar una capa superficial, transformándola en vidrio para obtener un aislante eléctrico (esto se consigue con facilidad calentándolo al vapor de agua); pueden imprimirse en él líneas de aluminio para que conduzcan la electricidad como si fuesen cables; o se le puede bombardear con átomos de yodo y otras impurezas, que penetran en él y lo hacen conductor de la electricidad en determinadas condiciones pero no en otras. A causa de esta propiedad se le denomina “semiconductor”.

La fabricación de transistores es muy simple. Requiere tan sólo los cuatro pasos representados en la figura 4. En primer lugar, se reviste el silicio con óxido (A). A continuación se abre un surco, dejando un puente en medio (B). Se quita el soporte (C) y se imprime una pista de aluminio a través del puente de óxido. Finalmente, el silicio que está descubierto se bombardea con átomos de yodo —el “dopante” (D).

El transistor tiene tres terminales, que se conocen tradicionalmente (y más bien de un modo inexacto) como puerta, fuente y drenaje. El objeto del ejercicio es controlar la circulación de corriente entre la fuente y el drenaje. La electricidad puede circular perfectamente bien a lo largo del silicio dopado. Pero aparentemente hay un obstáculo: el silicio situado debajo del aluminio no queda dopado, ya que se encuentra protegido de los átomos del dopante por el puente de aluminio y óxido. Así, debería ser aislante; no obstante, debido a los misterios de la física de los semiconductores, el trocito de silicio sin impurezas que queda debajo del puente conducirá la corriente si existe un campo eléctrico a su alrededor. Esto puede lograrse con cierta facilidad produciendo una tensión eléctrica en la puerta (en la pista de aluminio). Si se acciona la tensión, la corriente circulará de la fuente al drenaje (E). Si se desconecta, dejará de circular (F).

Aprovechemos esta propiedad conectando el transistor a un círculo sencillo. La fuente se conecta a un suministro de corriente de 5 voltios a través de un ‘resistor’. El drenaje se conecta a tierra (0 voltios). Si aplicamos tensión a la fuente, la corriente podrá circular de la fuente al drenaje.

Si el conjunto del circuito ha sido diseñado correctamente, la corriente circulará hacia fuera, a través del drenaje, más rápidamente que hacia

dentro, a través del resistor, y la fuente, por tanto, se encontrará a 0 voltios.

Si sacamos la tensión de la puerta, la corriente no circulará y la fuente se encontrará a 5 voltios. Y esto es precisamente lo que nos proponíamos consuir desde el principio: un interruptor controlado eléctricamente.

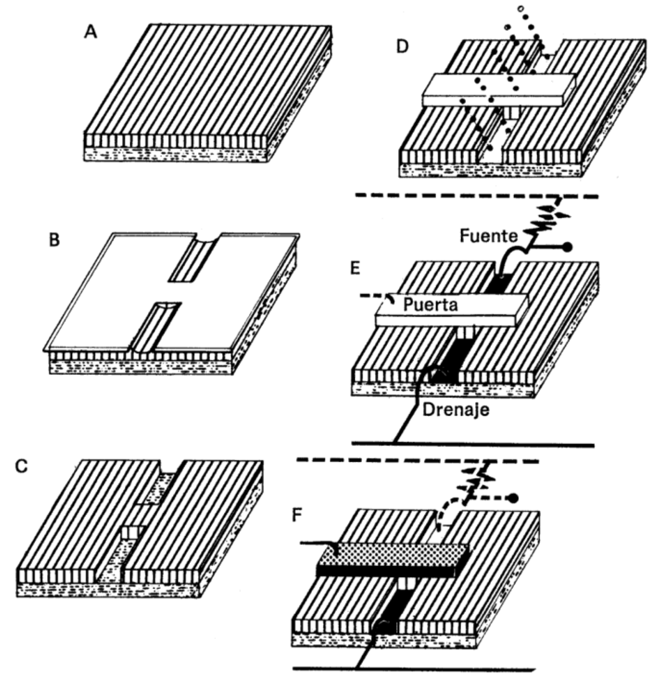


Fig. 4. Las distintas etapas en la construcción de un transistor.

Puertas

El transistor que acabamos de construir puede no parecer de gran utilidad. Pero tampoco un ladrillo es de gran utilidad por sí solo y, sin embargo, si tenemos los suficientes, podemos construir una casa o un rascacielos. Con los transistores ocurre exactamente lo mismo.

Para utilizar los transistores se les organiza en “puertas”, es decir, pequeños grupos de dispositivos que están diseñados para realizar operaciones lógicas de utilidad.

En la figura 4 ya hemos construido la puerta más simple: un inversor o puerta NOT (llamada así porque la salida no corresponde a la entrada; puesto que esta última sólo puede ser 0 o 1, el 1 se transforma en 0 y el 0 se transforma en 1). Se aplica una tensión de 5 voltios o ‘1’ lógico de entrada y se obtiene 0 voltios o ‘0’ lógico de salida. Y a la inversa, entremos 0 y obtendremos 1 de salida.

Esto es casi todo lo que se puede hacer con una entrada única. Si se tienen dos entradas y una salida y todas pueden ser o 0 o 1, se pueden hacer tres cosas: aplicar AND (Y) a las entradas; OR (O) a las entradas; o aplicar OR excluyente (XOR) a las entradas.

Estas tres operaciones se muestran en las “tablas de verdad” más abajo. En las dos columnas de la izquierda se indican las posibles combinaciones de las dos entradas A y B. En la columna de la derecha aparece la correspondiente salida. (La utilización de la palabra “verdad” proviene de los días en que únicamente los lógicos se ocupaban de estas cosas.)

NOT

A	NOT A
1	0
0	1

AND

A	B	A AND B
1	1	1
1	0	0
0	1	0
0	0	0

OR

A	B	A OR B
1	1	1
1	0	1
0	1	1
0	0	0

XOR

A	B	A XOR B
1	1	0
1	0	1
0	1	1
0	0	0

Una manera de enfocar estas operaciones es como si fuesen un test de las entradas. En la puerta AND, por ejemplo, si las dos entradas son 1, la salida es 1; en cualquier otro caso, la salida es 0. En la puerta OR, si las dos entradas son 0, la salida es 0; en cualquier otro caso, la salida es 1. En la puerta XOR, la salida es 0 si las entradas son iguales, y 1 si son distintas.

Puesto que un transistor único invierte la señal, las puertas más fáciles de construir son NAND y NOR (puertas AND y OR con sus salidas invertidas; se aplican las tablas anteriores con 0 y 1 intercambiados en las salidas).

La salida de la puerta NAND sólo puede ser 0 si las dos entradas son elevadas; en caso contrario es 1. La salida de la puerta NOR es 0 si las dos entradas son elevadas; en caso contrario es 1. Ambas puertas pueden construirse con unas tuberías y agua, tal como se indica en la figura 5. El agua sólo sale de la tubería AND si los grifos A y B están abiertos. El agua sale por la tubería OR si el grifo A o el B está abierto.

De nuevo nos encontramos con que estas operaciones por sí mismas no parecen de gran utilidad, pero consideremos las tres cosas que en las páginas 17-20 hemos visto que realiza un procesador de 8 bits. Fundamentalmente, tiene dos registros y una de las cosas que puede hacer es compararlos para ver si contienen el mismo byte de 8 bits.

REGISTRO A:	1	0	1	0	1	1	1	0
REGISTRO B:	1	0	1	0	1	1	1	0
XOR A,B								
REGISTRO C:	1	1	1	1	1	1	1	1

Si construimos un circuito que aplique la OR excluyente a los bits de los dos bytes tomados de par en par y a continuación aplique la OR al byte resultante por pares en cascada, obtenemos un único bit que es 1 si A y B son diferentes o 0 si son iguales:

```
A      1   1   0   1   0   1   1   0
B      1   1   0   1   0   1   1   0
XOR    0   0   0   0   0   0   0   0
OR       0       0       0       0
OR           0               0
OR                   0
```

Comparación: A y B iguales Probemos ahora con A y B distintos

```
A      1   1   0   1   0   1   1   0
B      1   1   1   0   0   1   1   0
XOR    0   0   1   1   0   0   0   0
OR       0       1       0       0
OR           1               0
OR                   1
```

Comparación: A y B distintos

Así podemos ver una aplicación incluso para un repertorio de puertas tan sencillo como éste. Podríamos casi diseñar un circuito integrado que hiciese la función de comparación de 8 bits de un procesador. Tratemos de sumar dos bits. A menudo, en los problemas lógicos resulta de gran ayuda empezar escribiendo todas las posibilidades:

A		B		Resp.	Lleva
0	+	0	=	0	0
1	+	0	=	1	0
0	+	1	=	1	0
1	+	1	=	0	1

La "respuesta" se produce al aplicar XOR a las entradas, y el "lleva", al aplicar AND a las mismas.

Podemos ampliar esta operación para sumar dos números de 8 bits

LLEVAN:	01111010
REGISTRO A:	00111010
REGISTRO B:	01011010
+	————
REGISTRO C:	10010100

Esto es ligeramente más complicado, pero todavía se puede hacer. Si trabajamos de arriba abajo, tenemos *tres* bits para sumar: el bit A, el bit B y el que lleva la suma anterior (que vale 0 en la columna de la derecha). Sin embargo, la suma en bits es igual a la que aprendimos en la escuela: A + B + C es lo mismo que A + B sumado a C. Así que podemos usar el sumador de dos entradas operando en la tabla anterior por etapas de dos en dos, con un OR para clasificar el lleva final.

Y, si se siente capaz de hacerlo, puede construir un circuito para la sustracción. Si se puede comparar, sumar y restar, también se puede multiplicar y dividir, y si se puede multiplicar y dividir, se pueden resolver ecuaciones, hacer estadísticas y predecir las consecuencias de complicados sucesos. Volviendo a las funciones lógicas, si se pueden realizar, se pueden combinar entre sí para llevar a cabo cualquier operación que pueda describirse como una serie de pasos lógicos. Computar es en realidad tratar de reducir operaciones humanas útiles a unos pasos rígi-

damente establecidos. Tal como veremos, esto está resultando sorprendentemente difícil. La gente es más inteligente de lo que pensamos.

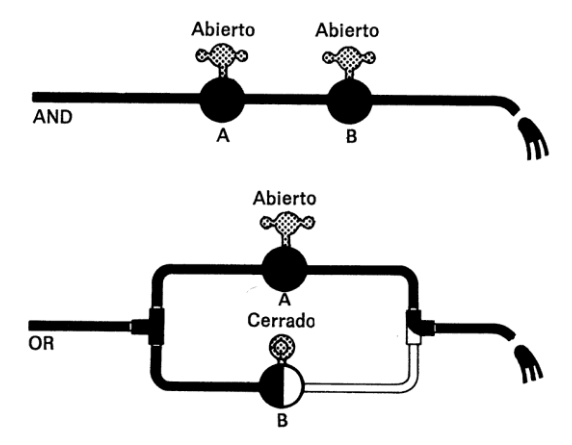


Fig. 5. Los principios de las puertas AND (Y) y OR (O) pueden entenderse fácilmente utilizando tuberías y grifos. Si en el diagrama superior los grifos A AND B están abiertos, el agua circula. Si uno de los dos está cerrado, el agua deja de circular. En el diagrama inferior, si está abierto el grifo A OR B, el agua circula.

BUSES

En las operaciones con computadores, a menudo es necesario conectar gran número de dispositivos entre sí. Podrían ser chips en la placa de un procesador, placas en un computador, computadores en una red local, o redes en un sistema nacional o internacional. También podría tratarse de radares y armas en un buque de guerra, controles en un avión de línea o robots, herramientas mecánicas y sensores en una fábrica automatizada.

La forma más sencilla de efectuar estas conexiones es unir cada subsidiario mediante un cable al controlador central formando una estrella. Existen, sin embargo, varios inconvenientes en este esquema: posiblemente habría que disponer de una enorme cantidad de cable y se necesitaría sin duda un enchufe extra en el sistema central para cada uno de los dispositivos exteriores. Si se añaden nuevos dispositivos (los sistemas de

computadores siempre se amplían), llegará un momento que se acabarán los enchufes.

La alternativa consiste en un *bus:* un conector que pasa por todos los subsidiarios del grupo, uno tras otro. La instalación eléctrica de las casas modernas está montada en anillos; cada anillo es un ejemplo sencillo de un bus que proporciona electricidad a las tomas de corriente. Los chips están conectados en un tablero de computador mediante un bus que transporta la energía eléctrica, los datos y las instrucciones por toda la máquina.

Cierto que esto podría presentar algún inconveniente. Los periféricos no necesitan actuar todos al mismo tiempo o hacer la misma cosa. Si todos los chips o dispositivos están conectados a los mismos trozos de cable, tiene que haber un controlador del sistema y algún modo de indicar a los periféricos el momento en que cada uno de ellos debe entrar en acción.

La esencia de una estructura en bus reside en el hecho de que aunque todo está siempre conectado al bus, cada dispositivo sólo toma información del bus, o pone información en él, cuando se le ordena que lo haga.

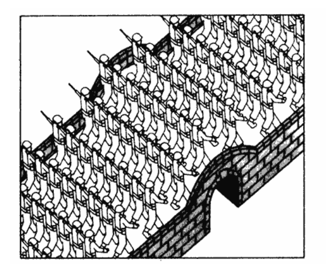
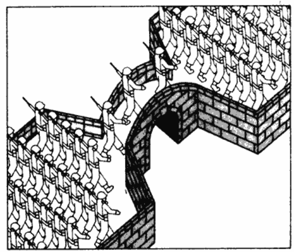



Fig. 6

Los buses que funcionan fuera del computador están conectados a él por “salidas” (véanse pp. 38-40) y se clasifican en dos categorías: en paralelo y en serie. La forma más sencilla de examinar el problema consiste en abordarlo como si se tratase de cruzar un río. Imaginemos un regimiento de soldados marchando en fila de a ocho (en paralelo) que llegan a un río (fig. 6). El proyectista tiene dos opciones para construir el puente: puede construir un puente ancho y caro que permita a los soldados cruzarlo en fila de a ocho; o bien puede construir uno más estrecho y barato que les permita pasar en fila de a uno (en serie). Al diseñador del bus se le presenta el mismo tipo de elección: tiene una serie de bits que

deben ir de una caja a otra en un cierto tiempo. Pueden enviarlos uno después de otro a través de un único cable o disponer cierta cantidad de cables en paralelo, de manera que sean varios los que puedan efectuar el trayecto del mismo.

Puesto que los datos viajan por el interior del computador en paralelo, un bus en serie debe salir de un chip especial (el SIO: entrada-salida en serie), que toma 8 bits en paralelo y los lee de uno en uno a la velocidad adecuada para pasarlos a cable y viceversa.

Esta operación precisa de más chips y aumenta el coste de fabricación del computador, pero disminuye el de las conexiones externas. Esta es la razón por la que los modelos de periféricos, tales como impresoras, que aceptan datos en serie cuestan más que los que los toman en paralelo.

Pero cuando el diseñador de un computador necesita transmitir datos a gran velocidad, por ejemplo, entre los discos y el procesador, utilizará un bus en paralelo.

Echemos una mirada a una oficina informatizada de un futuro cercano. Elena, una de las responsables, quiere obtener un archivo del disco para su edición. Su terminal contiene algún software permanente que sabe cómo controlar la red. Escribe un mensaje para ordenar lo que desea. Su computador espera hasta que el equipo central termina con lo que está haciendo y empieza a “sondear” las estaciones de trabajo, lo que realiza enviando una serie de mensajes a cada una de las estaciones por turno: «Número 1 — ¿Quieres algo?», «Número 2 — ¿Quieres algo?»...Cuando las otras estaciones de trabajo escuchan la llamada “Número 1...” se cierran, de manera que el campo central puede estar seguro de que cualquiera que sea la respuesta, ésta proviene de la Número 1 y de ninguna otra estación. Elena está en el Número 3 y, cuando llega el turno, envía su solicitud del archivo. El equipo central interrumpe su sondeo para enviar un mensaje al disco —que podría ser el número 63— y le dice que empiece a leer el contenido del archivo que el Número 3 pide y lo envíe al bus. Número 3 está a la escucha y, cuando el texto aparece, lo copia en la memoria donde Elena puede modificarlo o simplemente leerlo. Cuando el disco ha enviado todo el texto que el Número 3 desea, transmite una señal de “terminado el bus” y el equipo central continúa sondeando, lo que realiza paciente y eficazmente durante todo el día.

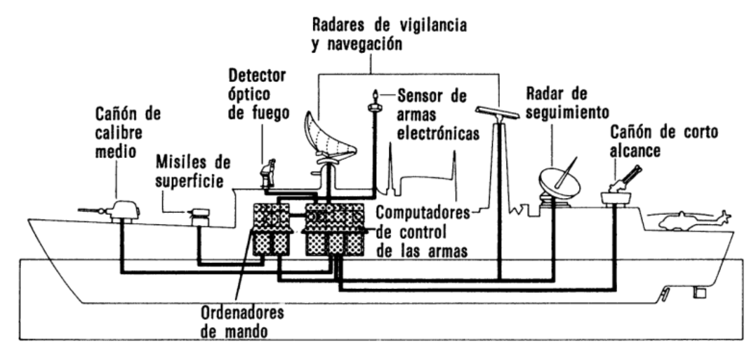


Fig. 7. Un buque de guerra es un buen lugar para un bus de datos. Un único hilo de fibra óptica (doblado o triplicado en previsión de posibles daños de combate) puede enlazar todos los sensores, computadores y armas del barco entre sí según las necesidades. El bus reemplaza kilómetros de cable, pesado y vulnerable, que de otro modo se necesitarían para enlazar en forma de estrella todas las partes. El bus de fibra óptica precisa de un mantenimiento mucho menor y es inmune a las interferencias eléctricas.

Todas las operaciones reseñadas parecen más bien pesadas, pero en la práctica ocurren tan rápido que los usuarios no se dan cuenta del proceso.

MEMORIA

Si el procesador es el motor de los computadores, la memoria es la sala principal, el espacio a través del que viajan los programas. A este tipo de memoria viva se le llama memoria de acceso directo *(random access memory* o RAM). "De acceso directo" significa que es posible leer o escribir directamente cualquier byte que contenga, sin tener que abrirse paso a través de otros materiales para acceder a ella. La memoria de contenidos fijos se llama memoria sólo de lectura *(read-only memory* o ROM). Entre estas dos memorias se encuentran diversos tipos de memoria programable sólo de lectura *(programmable read-only memory* o PROM).

Como ya se ha dicho, cuanta más memoria tenga un computador, mejor. Veremos más adelante que la cantidad de memoria que posee un computador depende de su precio, que a su vez depende de la densidad

de circuitería que los fabricantes de chips de memoria han introducido en sus pequeñas plaquitas de silicio.

Por el momento nos ocuparemos de cómo conseguir que los chips tengan memoria. En la página 11 vimos que un transistor es esencialmente un interruptor electrónico. Si se crea una diferencia de potencial en una puerta, permite que circule una corriente; si se suprime este voltaje, la corriente deja de circular. No hay en esto nada particularmente excitante. Pero si se conectan dos transistores de manera que uno controle al otro, se obtiene lo que los ingenieros electrónicos llaman un *flip-flop*.

En la figura 8 tenemos dos transistores con sus fuentes conectadas al carril positivo de 5 V a través de dos resistores, con sus drenajes conectados a tierra y con la puerta de cada uno de ellos conectada a la fuente del otro. Supongamos que se pone en funcionamiento el Tr 1. La corriente circulará a través de él con más rapidez que a través del resistor, de manera que su fuente está muy cerca de hallarse a 0 V. Como la puerta del Tr 2 está conectada a la fuente del Tr 1, Tr 2 se pone en funcionamiento, con lo que se eleva el voltaje en su fuente, ya que la corriente circula a través de su resistor pero no encuentra salida. Además, la puerta de Tr 1 está en funcionamiento por hallarse conectada a este elevado voltaje, de manera que el voltaje en la fuente de Tr 1 es bajo, tal como era justamente en un principio.

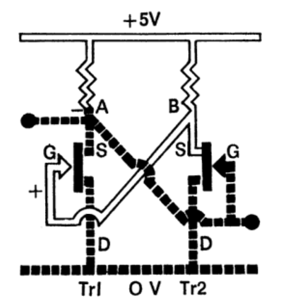


Fig. 8. Una célula de memoria “estática” formada por dos transistores. Uno está en funcionamiento y el otro no, almacenando así un 0 o un 1. (Aquí la salida en A es de voltaje bajo o 0). Están conectados entre sí de manera que cada uno de ellos mantiene el otro en el mismo estado hasta que se escribe un valor diferente en el circuito.

El resultado neto de todo esto es que el circuito se mantiene siempre en su estado inicial. El punto B siempre estará a voltaje elevado y el punto A bajo. Frente a cualquier cambio, los transistores interconectados actuarán en el sentido de restaurar las condiciones iniciales. Tenemos, por tanto, una memoria. Pueden disponerse otros circuitos en conexión con A o con B —no importa mucho con cuál de los dos—: el flip-flop

recordará un 0 o un 1, voltaje alto o bajo. Por supuesto, esto no sería de gran utilidad si no fuese posible cambiar lo que hemos almacenado en este dispositivo, es decir, si no fuese posible “escribirle”. De hecho, un voltaje lo suficientemente elevado en cualquiera de las dos puertas establecerá un nuevo flip o un nuevo flop. Por tanto, además de una salida, por ejemplo, de A, tendremos que tener una entrada en la puerta del Tr 2. Un voltaje bajo o nulo mantendrá A bajo; un voltaje elevado hará y mantendrá A elevado. Un circuito como el descrito puede construirse fácilmente con transistores y resistores individuales. Un diseñador de chips reemplazaría los resistores (que son difíciles de construir) por transistores con su drenaje conectado a su puerta.

Con un flip-flop puede conservarse un bit de información; para conservar una cantidad de información útil, se imprimen conjuntamente varios miles de estos circuitos en la misma ficha de silicio, obteniéndose así un chip. Este tipo de RAM recibe el nombre de “estática”, porque los bits permanecen inalterados, a diferencia de lo que ocurre con la RAM “dinámica” de la que hablaremos en seguida.

Cuatro transistores ocupan cuatro veces más espacio que un transistor, lo que significa que el fabricante de chips puede conseguir con un chip sólo una cuarta parte de la memoria que conseguiría si utilizase un solo transistor. Puede ser más interesante una célula de memoria que utilice sólo dos transistores; lo que se obtiene con un transistor controlado por su puerta. La puerta es simplemente una pista de aluminio extendida sobre una capa aislante de óxido de silicio; y está eléctricamente aislada, de manera que, al menos en teoría, cuando se carga eléctricamente, la carga no se pierde. Permanecerá allí cumpliendo la tarea de controlar el flujo de corriente desde la fuente al drenaje.

Tenemos otra vez una memoria. Podemos decir lo que se ha escrito en la puerta —0 o 1— mirando si circula corriente de la fuente al drenaje.

En la práctica, los electrones no permanecen quietos en la puerta mucho tiempo; de hecho, no más de una milésima de segundo. Pero en este tiempo el procesador central puede realizar varios miles de operaciones: es un tiempo lo bastante largo para permitir la realización de una buena cantidad de trabajo. En una milésima de segundo el procesador puede recorrer todas sus células de memoria, leer lo que en ellas está escrito y reescribirlo para que permanezca allí durante otra milésima de segundo.

A este proceso se le llama “refrescar la memoria” y está a cargo de circuitos especiales ligados a los chips de memoria.

Cualquier chip de memoria necesita un determinado número de clavijas para ser conectado con el resto del computador, y este número depende íntimamente de la cantidad de memoria que contiene. Debe tener suficientes líneas de direccionamiento para llegar a todas sus células de memoria, una línea de lectura/escritura que indique si llegan datos que deben ser almacenados o leídos y una línea de datos conectada a la salida de la célula particular. Por tanto, si un chip tiene 2^n células de memoria, necesitan *n* líneas de direccionamiento. Un clip de RAM de 64K [1], necesita que éste tenga 16 clavijas de direccionamiento.

Para recordar 1 byte de información, que es la unidad útil más pequeña, se necesitan 8 bits, u ocho de estos circuitos. Convendrá guardar cada uno de los 8 bits del byte en un chip distinto, con la línea de datos de cada chip conectada a una de las ocho líneas del camino principal de datos.

Existen otros varios tipos de memoria que se utilizan corrientemente. La ROM (memoria sólo de lectura) es mucho más simple; los datos se escriben una sola vez, cuando se fabrica el chip, y no pueden ser alterados posteriormente. Cada célula consiste en una conexión entre la línea de datos y el carril de energía de signo + o –, y esto se consigue imprimiendo una máscara durante la fabricación. Un chip de ROM puede almacenar volúmenes bastante grandes de programa o de datos, y puede aparecer ante el procesador simplemente como una sección de RAM que ha sido cargada con programa o datos. La mayoría de los microcomputadores tienen unos cuantos kilobits de ROM para el control de sus teclados y pantallas: una sección del código generalmente llamado “monitor” (que no debe confundirse con el monitor de vídeo, semejante a una televisión, en el que se realiza la visualización en pantalla).

El inconveniente del ROM reside en que deben fabricarse series de varios miles para compensar el coste de la máscara que es muy elevado. Una alternativa que aumenta el coste por chip, pero permite a industrias pequeñas fabricar cada chip cuando lo necesitan, es el PROM (memoria programare sólo de lectura). Se consigue a partir de un chip en blanco,

---

[1] “K” no significa 1.000 en informática sino 1.024; ya que 1.024 es 2^{10}. En consecuencia, $64\text{ K} = 64 \times 2^{10} = 2^6 \times 2^{10} = 2^{16}$.

escribiendo en él los datos o el programa, estableciendo o no un vínculo en cada ubicación de memoria. Esta operación se realiza mediante la aplicación de un voltaje elevado y, para mayor garantía, se somete a la supervisión de un computador. Un chip con exigencias aún menores pero más caro es el EPROM, memoria borrable sólo de lectura *(erasable programmable read-only memory).* Es algo así como una especie de RAM de larga vida, que permite programar las células una vez y puede recordar estos datos sin necesidad de que se le refresque la memoria.

EL TECLADO

El usuario medio de un computador pasa la mayor parte del tiempo en contacto físico con el teclado. Su teclado, con sus cerca de cincuenta teclas le permite realizar una amplia variedad de operaciones. Puede mecanografiar un texto como si se tratara de una máquina de escribir; puede entrar números para realizar cálculos; puede controlar el cursor en la pantalla (hacerlo subir, bajar, mover lateralmente); puede jugar a "Invasores del espacio", escribir poesía, hacer dibujos, etc.

Los teclados no son todos iguales; sin embargo, suelen presentar:

Las letras del alfabeto, con una tecla de cambio a mayúsculas y otra de fijar mayúsculas

Una barra espaciadora

Los símbolos £ $% & #

Un conjunto de paréntesis, llaves y corchetes (){}[]

Los símbolos aritméticos + – • / ↑ < >, es decir, de sumar, restar, multiplicar, dividir, elevar a una potencia, menor y mayor que

Los signos de puntuación ,;:.?!

Dos tipos de comillas ' y "

Algunos símbolos que son corrientes en los computadores, pero que se encuentran raramente en las máquinas de escribir: {\ ~

Cuatro teclas de control del cursor para moverlo arriba, abajo, a la derecha y a la izquierda

Además de estas teclas siempre se encontrarán al menos cuatro teclas especiales: DELETE, CONTROL, ESCAPE y RETURN. Como son muy importantes, vamos a explicarlas una por una.

DELETE (eliminar, anular) hace lo que nos gustaría poder hacer con la máquina de escribir: borrar el carácter que acabamos de pulsar erróneamente.

CONTROL o CRTL no muestra un carácter en la pantalla. Siempre se pulsa *con otra tecla* y cambia el significado de ésta (véase caracteres ASCII, p. 291). En efecto, manteniendo apretada la tecla CTRL se obtiene un segundo teclado. Los caracteres CTRL se escriben normalmente así: '↑A' significa el efecto producido al pulsar conjuntamente CTRL y A.

ESCAPE (cancelación): es utilizada en algunos paquetes de programas para cambiar de una modalidad de operación a otra.

RETURN (volver, retorno: en ocasiones denominada NEWLINE o ENTER) es la tecla más utilizada. Se diseñó originariamente para que tuviese el mismo efecto que se obtiene al pulsar la tecla de retorno del carro en una máquina de escribir: mueve el cursor en la pantalla, o la cabeza impresora sobre el papel, hacia el margen izquierdo y una línea más abajo. Para hacer esto envía *dos* caracteres: CARRIAGE RETURN (retorno de carro) seguido de LINE FEED (avance de línea). Pero RETURN ha adquirido un nuevo significado: «ir a buscar» o «ejecutar». Muy a menudo, cuando un programa le pide que escriba un nombre o un número, no piensa que ha terminado hasta que pulsa RETURN.

Algunos computadores tienen un sector del teclado separado, en donde se encuentran las teclas de números como en una calculadora, lo que es más cómodo para entrar series de números. Es también normal encontrar una fila de "teclas programables" cuando se pulsa una de ellas, se transmite una serie de caracteres desde una pequeña sección de RAM al teclado. Las teclas de función o programables pueden programarse para que realicen cosas muy complicadas. Por ejemplo, en el programa de tratamiento de textos que he utilizado para escribir este libro, se pulsa 'ESR↑R' para empezar una operación de "búsqueda y sustitución". Si mi máquina hubiera tenido un conjunto de teclas "programables" o de "función especial", podría haber almacenado esta secuencia y pulsar sólo una tecla para transmitir el efecto de tres. Cuando finalicé con el tratamiento de textos podría haber cambiado a otro paquete y dado funciones nuevas a las teclas programables. Los teclados con teclas programables tienen normalmente una ranura que acepta fichas de cartón que llevan escritos sus significados actuales. En algunos sistemas el teclado y la pantalla van

unidos, de manera que el programa puede escribir las anotaciones de las teclas de función en la línea inferior de la pantalla.

Un teclado aceptable debería tener una tecla que permitiera al usuario seguir tecleando incluso cuando el computador no puede aceptar pulsaciones de las teclas debido a que está haciendo otra cosa en aquel momento; el teclado almacenará unas pocas en una pequeña sección de RAM y las enviará cuando las condiciones hayan mejorado. La mayoría de los teclados actuales imitan a las máquinas de escribir eléctricas: si se mantiene una tecla apretada se repite el carácter.

Un refinamiento que no es necesario es el chasquido de la tecla. Algunos fabricantes hacen grandes esfuerzos para imitar el ruido de una máquina de escribir, poniendo un altavoz en el teclado que produce pequeños chasquidos. Esto puede ser de utilidad para señalar las equivocaciones; si el computador envía a la pantalla "↑G", debería producir, entonces, un corto "mip" electrónico en el teclado.

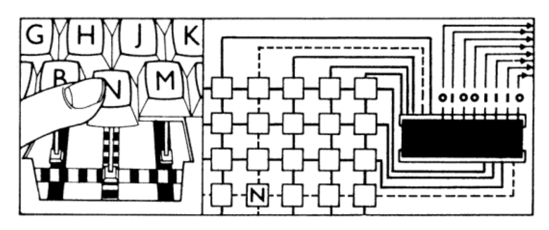


Fig. 9. Al pulsar una tecla en el teclado de un computador se provoca un contacto entre un conductor horizontal y otro vertical. Ambos están conectados a una memoria sólo de lectura (ROM) que traduce los dos cables conectados al código ASCII correspondiente a esta letra. En este caso, la letra es 'N' y el código es 01001110, o 78 en sistema decimal.

Ya vimos en las páginas 13-15 que todos estos caracteres del teclado están codificados como caracteres ASCII (en la p. 291 se encuentra la tabla completa). El código ASCII es una de las ideas menos simples de la industria informática y tiene algunos rasgos muy inteligentes. Obsérvese que 'A' (65) es 32 menos que 'a' (97). De manera que 'B' (66) es también 32 menos que 'b' (98). Para convertir las letras mayúsculas en minúsculas basta con añadir 32. '↑A' es 1, de manera que los caracteres de control están codificados en ASCII restando 64 al código alfabético.

Obsérvese que los números aumentan a medida que recorremos el alfabeto de la 'a' a la 'z'. Así para clasificar en orden alfabético basta con

clasificar los códigos ASCII en orden numérico. Lo mismo se aplica a los caracteres numéricos: '1' (49) es menor que '2' (50), de este modo, el mismo programa que clasifica las letras en orden alfabético también clasificará los dígitos ordenadamente.

Por encima de 127, el conjunto de caracteres ASCII tiende a agotar sus posibilidades. Muchos microcomputadores de aprendizaje proporcionan distintas variedades de caracteres "gráficos" (cuadros, puntos y estructuras) a los que se puede acceder modificando el teclado. Las mejores máquinas de oficina se sirven de este espacio ante la existencia de algunos caracteres europeos especiales, tales como á y ü.

CONECTORES DE SALIDA

Por sí solo, un computador es un objeto bastante inútil. Tendrá un interruptor de abrir/cerrar y, con suerte, una luz que indique si está en marcha o apagado.

Para que sea útil debe estar conectado al menos a un teclado, a una pantalla y a una impresora; y muy a menudo también a otros elementos: palancas de mando para controles de videojuegos, bolas de mando o ratones para mover el cursor, o paneles gráficos para dibujar. Si se desean salidas más exóticas que las que proporciona una impresora ordinaria, deberá conectarse un *plotter* (dispositivo trazador de gráficos), un láser, un robot o cualquier otro dispositivo imaginable.

La utilización de todos estos instrumentos resultaría imposible si cada uno de ellos necesitase un tipo particular de conexión. Para simplificar, se ha inventado otro conjunto de conexiones (casi) estandarizadas llamadas *ports* ("puertos", que hemos traducido como "conectores de salida"). (Probablemente se les denomina puertos, por analogía con los puertos marítimos o aeropuertos de un país, ya que son caminos de y hacia el mundo exterior.) Básicamente existen dos tipos de conectores de salida: en serie y en paralelo. Tal como vimos en las páginas 28-31, una salida en serie envía o recibe los bits de un byte de uno en uno a través de dos cables; una salida en paralelo envía o recibe ocho o más a la vez a través de tantos cables como bits.

Si se observa un conector de salida en la caja del computador, lo que se ve en realidad es un enchufe de 25 clavijas (normalmente hembra, aunque no siempre). Para utilizarlo, usted (o la persona que diseñó el

periférico que se quiere conectar) debe saber si es en serie o en paralelo y si se ajusta a uno de los estándares (RS 232 en serie, Centronics o IEE 488 en paralelo [también existen otros estándares y, probablemente, el principiante no será capaz de enfrentarse con los pormenores de cada uno de ellos]). Además, si la salida es en serie, debe conocer la velocidad con que se supone que llegarán los datos (esta velocidad se mide en baudios: bits por segundo) y si lo hace regularmente o se supone algún otro “protocolo”.

Si el periférico se encuentra a cierta distancia —quizá metros, quizá cientos de kilómetros— es importante comprobar la integridad de los datos a su llegada. Este es el momento en el que el “bit de paridad” descrito en las páginas 13-15 entra en juego, al igual que otros procedimientos mucho más sofisticados para asegurarse de que todo ha funcionado correctamente durante la transmisión. El tema de las comunicaciones en computadores es muy interesante de por sí y los libros sobre este tema llenan secciones enteras de las bibliotecas.

Dentro del computador, cada conector está dispuesto de modo que se presente al procesador como dos posiciones de memoria: una para los datos que han de enviarse o recibirse y otra para que indique el camino que siguen (byte de “datos” y byte de “situación”). Esto simplifica la tarea del procesador. Si está enviando datos a través de un conector, sólo tiene que mirar ocasionalmente el byte de situación para ver si se necesitan más datos. Si es así, se detiene el programa en marcha y el procesador escribe más datos para el conector de salida. Cuando la unidad externa señala que ya tiene bastantes, el byte de situación cambia y el procesador para de escribir datos para el conector, prosiguiendo a partir de entonces con el trabajo encomendado.

Existen dos modos de controlar esta división de tareas: uno consiste en utilizar el dispositivo de interrupción del procesador. El cambio de 0 a 1 de un bit en el byte de situación para solicitar más datos pone una señal en una clavija especial del procesador. Esto le hace arrinconar lo que está haciendo y saltar a un programa distinto que, en este caso, le obliga a enviar datos hacia el conector. Una vez realizado este envío, cesa la interrupción y el procesador vuelve al programa que estaba desarrollando. No cabe la menor duda de que las interrupciones también pueden utilizarse para otras muchas cosas.

El segundo sistema de control es el *polling* (“sondeo”). En este esquema el programa principal hace que el procesador mire de vez en cuando al conector para ver si necesita más datos. En el proceso puede ir a diferentes periféricos, pidiéndoles a cada uno si requieren atención. Este es un buen sistema en los computadores que se supone que recibirán muchas entradas del teclado, ya que la máxima velocidad con la que el usuario puede pulsar el teclado es de una tecla por décima de segundo. Así, mientras el procesador realice todos los sondeos (incluyendo el del teclado) a una velocidad superior, el usuario pensará que obtiene toda la atención exclusiva del procesador. Y puesto que funciona a varios millones de ciclos por segundo, puede realizar gran número de cómputos en este tiempo.

PANTALLAS

La gran mayoría de las pantallas de los computadores son de tubos de rayos catódicos (*cathode-ray tubes* o CRT), similares a las de los televisores. Pero es posible que en el futuro se popularicen las pantallas de estado sólido; de ellas hablaremos más adelante. Un CRT es un objetivo sorprendentemente complicado y, si no fuera por la inmensa popularidad de la televisión, su fabricación resultaría muy cara.

Un CRT consiste en una botella de vidrio de fondo plano en la que se ha hecho el vacío. La base plana está tapizada con una especie de sal de fósforo. En el cuello hay un cañón que dispara un fino haz de electrones a la pantalla. Al incidir el haz en la pantalla, se produce un punto brillante. Dispositivos electrónicos de control pueden mover el haz arriba y abajo, a derecha e izquierda, aplicando el voltaje apropiado a dos pares de placas. De este modo es posible dibujar en la pantalla. Como el fósforo continúa brillando durante unas milésimas de segundo después de haber recibido el impacto del haz electrónico, es posible dibujar en toda la pantalla antes de que desaparezca la imagen. Se puede conseguir que este punto se mueva más de 40.000 km/h a través de la pantalla, lo que posibilita que el proceso pueda repetirse muchas veces en un segundo, creando así la ilusión de una imagen continua y sin destellos.

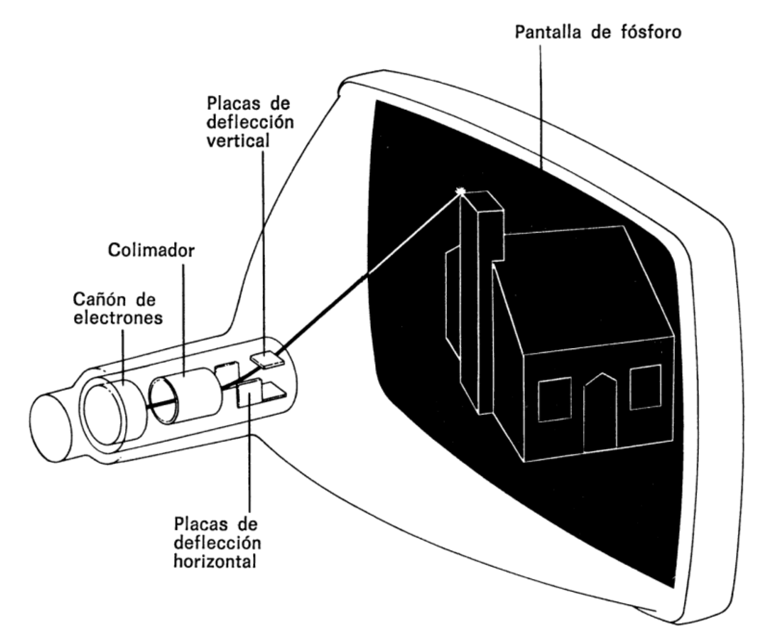


Fig. 10. Interior de un tubo de rayos catódicos (CRT). Un cañón, que contiene un alambre al rojo, dispara un haz de electrones. El haz pasa a través de un colimador que lo enfoca y, a continuación, entre dos pares de placas deflectoras que lo desvían a la izquierda o a la derecha, arriba o abajo mediante campos eléctricos. Atraviesa entonces un vacío hasta que incide en la capa de fósforo, que reviste el interior de la pantalla, donde deja un punto brillante.
El cañón puede disparar o no y el haz puede ser dirigido a cualquier punto de la pantalla para dibujar una imagen o escribir, bien un texto bien números. Cada 1 /25 segundos debe reescribirse la imagen.

Existen otras dos maneras de dibujar en la pantalla. En una de ellas, conocida como gráficos de “vector”, se utiliza el haz como si fuese un lápiz, y se dibujan de hecho formas sobre el fósforo. Este método proporciona resultados de gran calidad, pero es muy lento.

El otro método (utilizado en la mayor parte de los microcomputadores) es conocido como *raster sean*, ya que, al igual que en la televisión, el haz recorre *(scans)* la pantalla de lado a lado en una red de líneas paralelas. Cuando el punto de incidencia del haz cruza las líneas en la ima-

gen, la circuitería de control lo enciende y apaga una y otra vez. Dibuja la imagen mediante una serie de puntos que, en una pantalla de buena calidad, se encuentran lo suficientemente próximos unos de otros para dar la sensación visual de continuidad en la imagen.

Sería interesante considerar cada punto como una unidad de imagen independiente, que podría ser encendida o apagada a voluntad. Entre los profesionales tales imágenes se conocen con el nombre de *pixel* —contracción de *picture cell* (puntos de imagen). Esto significa que se debería dar a cada punto por lo menos un bit de RAM. Sin embargo, existen alrededor de 600 × 600 = 360.000 puntos en una pantalla de televisión —y, ni siquiera con ese número, se consigue una imagen realmente definida, pero, aun así, para conseguir colores y sombras aceptables se necesita al menos un bit por pixel. Esto supone que sólo para controlar la pantalla se necesitaría medio megabyte de RAM, lo que excede la capacidad de una máquina de 8 bits y ocuparía una buena parte de la memoria disponible en una máquina de 16 bits.

Una máquina de 8 bits controla su pantalla dividiéndola en menos pixeles de mayor tamaño. El esquema estándar considera la pantalla como formada por 24 líneas horizontales y 80 verticales. En cada uno de estos 1.920 cuadros del computador dibuja un número preestablecido de imágenes, cada una de ellas gobernada por el número ASCII (véanse pp. 13-15). Como estos números ASCII están almacenados en un byte único, sólo se necesitan 2 kilobytes de RAM.

El código ASCII de la ‘A’ es 65, o el byte ‘01000001’. Pero con esto todavía se está muy lejos de conseguir la forma ‘A’. Lo que la máquina hace es almacenar las formas como patrones de puntos, normalmente 5 en sentido horizontal y 9 en el vertical, en la memoria sólo de lectura. Cuando tiene que dibujar una ‘A’, sabe que la primera línea tiene que ser: apagado, apagado, encendido, apagado, apagado. La línea siguiente tiene que ser: apagado, encendido, apagado, encendido, apagado. Y así con las restantes líneas. Cuando la pantalla tiene que escribir una línea completa de texto, debe saber antes de empezar cómo será la fila superior de puntos para todos los caracteres contenidos en la línea; después, cómo será la segunda fila y así sucesivamente.

En algunas pantallas utilizan pixeles de sólo siete puntos verticales, que resultan insuficientes para mostrar las formas de letras como la ‘y’ y la ‘p’, que poseen fuertes pendientes.

Un tipo completamente distinto de pantallas está formada por las que utilizan tecnología del estado sólido. Forman sus imágenes con diodos emisores de luz y con cristales líquidos. Cada punto de la pantalla está bajo el control de dispositivos electrónicos. En la pantalla de diodos, cada punto es una minúscula fuente de luz que se enciende o apaga como una bombilla eléctrica. En la pantalla de cristal líquido, los puntos son áreas de líquido que se hacen transparentes u oscuras sometiéndolas a un determinado voltaje. Cada punto tiene un par de electrodos transparentes situados en ángulo recto que pueden conectarlo o desconectarlo.

Estos dispositivos de visualización podrían resultar mucho mejores que los anticuados CRT, ya que son de menor tamaño, de fabricación más económica y consumen menos energía. En la práctica, resultan muy difíciles de fabricar, puesto que para que se consiga un buen nivel de resolución, se necesitan muchos puntos y mucha electrónica de control. Además, presentan problemas en relación con la estabilidad, la temperatura y el consumo de energía.

GRÁFICOS

Ya hemos visto cómo se forman las imágenes en la pantalla; la pregunta que nos hacemos ahora es: ¿Las imágenes de qué?

Un grupo de formas está evidentemente constituido por las letras y números del teclado. Los microcomputadores baratos, que están diseñados para utilizar la pantalla del televisor, tienen 40 caracteres o pixeles de extremo a extremo de la pantalla; las máquinas de precio más elevado que poseen un monitor disponen de 80 caracteres. La calidad de las letras tiende asimismo a ser mejor.

Con independencia de que la pantalla posea 40 u 80 caracteres de extremo a extremo cada letra ocupa el mismo espacio. Esto significa que, al igual que en una máquina de escribir, las ‘i’ tienen mucho espacio libre alrededor, mientras que las ‘w’ y las ‘m’ están un poco apretadas. Una máquina bien diseñada tendrá, por supuesto, mayúsculas y minúsculas y las minúsculas, tales como la ‘y’ y la ‘g’, tendrán sus perfiles bien dibujados. Dado que la mayoría de los usuarios de computadores del mundo hablan y escriben inglés, el conjunto de caracteres que se emplean tiende

a ser el británico o el norteamericano (la diferencia entre ambos reside en los signos del guión y de la libra, que son iguales en el código ASCII).[2]

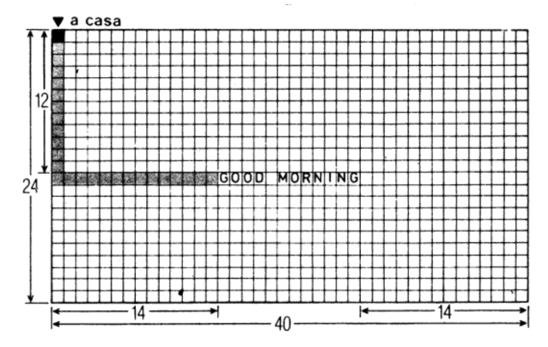


Fig. 11. El proceso para escribir “GOOD MORNING” en el centro de una pantalla de 40 columnas y 24 filas, no es tan simple como pudiera parecer. Para situar la frase en posición correcta, debe efectuarse un cálculo que, por supuesto, puede ser realizado por el programa. Primero se envía el cursor “a casa”, es decir, al extremo superior izquierdo. A continuación, se imprime 12 avances de línea para situarse en la línea central. “GOOD MORNING” tiene 11 letras y un espacio, de manera que se necesita imprimir (40–12)/2 = 14 espacios para llevar el cursor a la posición correcta para empezar.

Sin embargo, los mercados de los países del norte de Europa están creciendo lo suficiente como para que resulte interesante para los fabricantes proporcionar determinados caracteres especiales que se necesitan en diversos idiomas, tales como las ‘e’ y ‘a’, acentuadas en francés y las diéresis en alemán. Estos caracteres tienen que teclearse separadamente ya que el mecanismo de la pantalla no permite que el cursor retroceda y añada un elemento extra, tal como un acento, a un carácter ya existente. Es bastante divertido abrir estas máquinas y encontrar dentro un “rústico” interruptor que hace entrar en acción los bits de ROM apropiados para las necesidades locales.

---

[2] La razón por la cual los caracteres de la mayoría de los computadores corresponde a la lengua inglesa es debida a los países de origen de la tecnología utilizada y al código ASCII normalizado. Existen computadores adaptados a la lengua castellana que incluyen la “ñ” por ejemplo, en los que se ha sustituido alguno de los signos especiales por este carácter.

Si desea obtener letras de mayor tamaño que el normal, tendrá que formularlas del modo siguiente:

```
P P P P P P P P P P
  P P P           P P
  P P P             P P
  P P P           P P
  P P P P P P P P P
  P P P
  P P P
  P P P
  P P P
P P P P P
```

Existen máquinas con conjuntos de caracteres *(character sets)* de diferentes tamaños, de manera que resulta posible escribir en la pantalla tanto líneas de encabezamiento como tipos de imprenta. En principio, no hay ninguna razón por la que una máquina de 16 bits, con la memoria y capacidad de procesamiento de que dispone no pueda ofrecer en la pantalla un texto proporcionalmente espaciado, es decir, una visualización que dé a las letras un espacio proporcional a su anchura.

No hay nada intocable en las aproximadamente 120 formas que nosotros reconocemos como letras, números y signos de puntuación. El computador podría dibujar cualquier forma que no resultase excesivamente complicada para la estructura de puntos de su pantalla. Esto significa que el árabe, por ejemplo, no representaría un problema real. Las formas no son más complicadas que las del alfabeto inglés y el número de caracteres es aún más reducido, de manera que pueden ser almacenados en ROM y direccionados con códigos de un solo byte como el ASCII. El japonés presenta más dificultades ya que el katakana, la más simple de las dos modalidades de japonés, tiene alrededor de 2.000 caracteres y muchos de ellos son considerablemente más complicados que las letras del alfabeto inglés. Pueden, sin embargo, manejarse en una pantalla con células de caracteres de gran tamaño y con un código de direccionamiento de 2 bytes (1 byte sólo puede servir para representar 256 caracteres distintos).

Las letras y los números están lo bastante estandarizados y son tan necesarios como para que compense incorporarlos a la ROM. Muchos

computadores destinados a los aficionados a los videojuegos proporcionan asimismo un *grafics set, o* conjunto de diversas formas del mismo tamaño que las letras, que pueden ser utilizadas por los más decididos como sistema para la obtención de figuras y monstruos adecuados a sus juegos. Sin embargo, el programador quizá desee conseguir formas más complicadas, en cuyo caso tendrá que crearlas especialmente en RAM.

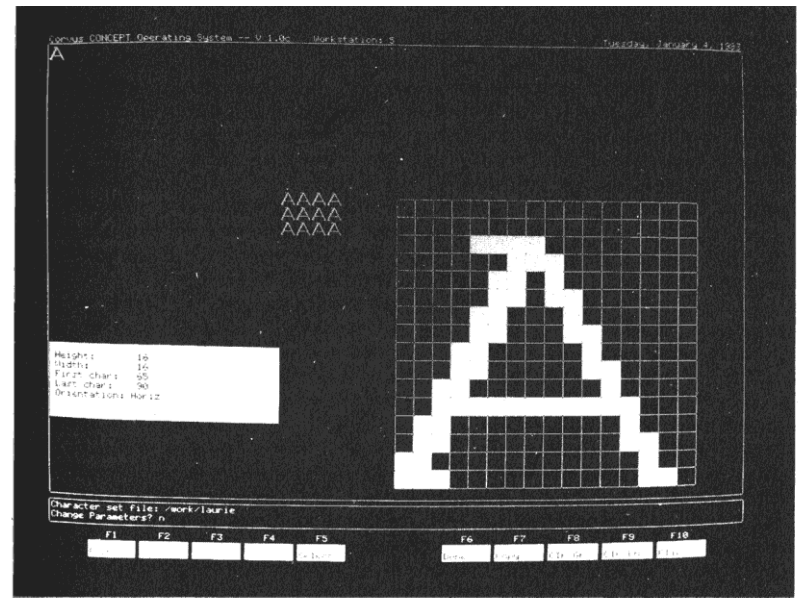


Fig. 12. Algunas máquinas de 16 bits permiten a los usuarios redibujar las letras de su pantalla. En estas máquinas, las letras, en vez de estar almacenadas permanentemente en ROM como en las máquinas de 8 bits, se guardan en un disco y se leen cada vez que se conecta el computador. En la fotografía puede verse un programa dando a una ‘A’ una nueva forma.

En todas las operaciones con gráficos, la pantalla es redibujada —normalmente 25 veces por segundo— por un área especial de memoria que formará parte de la RAM principal o estará separada. Esta área se denomina con frecuencia el “vídeo mapa”, porque cada byte o dos bytes que contiene corresponden a un pixel de pantalla (véanse pp. 40-43). En los computadores personales actuales, la pantalla tiene a menudo una

resolución de 200 × 400 pixeles más o menos, de manera que el vídeo mapa tiene que dirigir 80.000 pixeles. Cada pixel puede hacerse corresponder con un bit si se trabaja con un solo color y sin matices de tono. Esto exige 10 K de RAM. Cuatro colores (rojo, verde, azul y blanco, que corresponde al pixel "apagado") exigen 2 bits (ya que $2^2 = 4$) y 20 K de RAM. Todo lo que el procesador pone en el vídeo mapa automáticamente aparece en la pantalla la próxima vez que es redibujada. Como este redibujado ocurre con frecuencia, es posible producir efectos de animación en las imágenes de la pantalla.

Lo que debe recordarse es que cada vez se redibuja toda la pantalla. Este hecho tiene dos consecuencias. En primer lugar, para producir efectos de animación, la nueva pantalla tiene que ser exactamente igual que la anterior excepto en los detalles que se han "movido". Por ejemplo, si la imagen es la del pato Donald hablando, todo debe permanecer igual menos su pico. Cada 1/25 segundos aparece una nueva imagen con el pico en una posición ligeramente distinta. Este es el principio de las películas de dibujos animados, excepto en el hecho de que todos los "planos" están dibujados sobre el mismo trozo de "celuloide": la pantalla.

En segundo lugar, el computador debe ser capaz de calcular los cambios necesarios y reescribir toda la imagen para el vídeo mapa en 1/25 segundos. Si se tiene en cuenta que incluso el visualizador de baja calidad puede tener alrededor de 8 KB de vídeo mapa, esto supone una limitación drástica, lo que significa que o bien los cambios son simples o se utiliza un computador de mucha potencia. La tercera posibilidad en la realización de películas con ayuda de computadores consiste en emplear mucho más de 1/25 segundos en la renovación de la imagen, filmar los planos uno a uno, y después pasar la película a la velocidad apropiada.

A pesar de todo lo dicho, son muchas las cosas que pueden hacerse con la simple animación en un computador bidimensional. El problema principal es la renovación del vídeo mapa. En efecto, cada uno de sus bytes podría computarse cada vez. Para la animación tridimensional es necesario hacerlo así, lo que comporta una pérdida importante en la capacidad de procesamiento; sin embargo, para la animación bidimensional en computadores personales hay algunas soluciones sencillas. Existen muchos tableros de gráficos de alta resolución en el mercado; algunos se venden como accesorios, otros están incorporados a las máquinas, pero todos tienden a ofrecer el mismo tipo de posibilidades.

En general, es posible definir de antemano un determinado número de formas que pueden escribirse en la pantalla en cualquier posición. Los fabricantes de la máquina proporcionan las más elementales, las letras y los números (que pueden ser giratorios). También puede disponerse de determinados “caracteres gráficos”, que ocupan el mismo espacio que una letra estandarizada y consisten en cuadros claros y oscuros en combinaciones diversas. Si los combina una persona imaginativa podrá formar con ellos dibujos rudimentarios.

Quizás exista también un lenguaje de gráficos como el Logo (véase p. 98), u órdenes para gráficos como extensión del BASIC residentes (véanse pp. 86-97), que controlan el movimiento del cursor en la pantalla, trazan una línea entre dos puntos, dibujan círculos u oscurecen áreas. A veces se suministra un paquete de programas para hacer todo lo anterior.

Si se han incorporado colores al hardware, existe casi siempre alguna manera de definir una gama de 16 o 256 colores que pueden ser identificados con un número. Generalmente, esta gama se establece al inicio del programa, mediante la elección de los colores entre una gama muy amplia. Supongamos que todas las caras de las personas, en una serie de animación, estén coloreadas con el color 6. Al principio se asignaría al 6 un tono rosa. En un determinado momento aparece un fantasma y todo el mundo se vuelve gris de miedo. Este efecto puede obtenerse con un solo movimiento, simplemente cambiando el 6 del rosa al gris. Existe otro método que se basa en la utilización de tres planos de color, correspondientes al rojo, verde y azul, que pueden introducirse o no, dando así un total de ocho posibilidades de color.

Dos son los principales sistemas de ayuda al animador. El primero es conocido como *paging* y se basa en el uso de dos o más vídeo mapas. Se dibuja una imagen en uno de ellos (empleando, quizá, más de los 1/25 segundos de que en principio se dispone) y se le da entrada cuando ya está lista la siguiente salida de vídeo, mientras se prepara la siguiente imagen en una de las otras áreas. Esto resulta caro en términos de RAM.

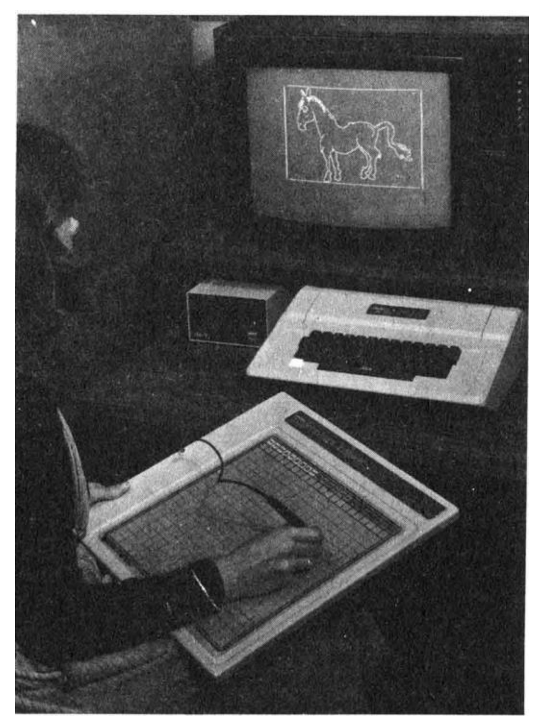


Fig. 13. Aunque es posible dirigir el cursor por la pantalla mediante un teclado, no es éste el sistema más artístico para dibujar. Un tablero de gráficos supone una mejora. El usuario dibuja con un lápiz electrónico, que puede llevar un lápiz convencional incorporado. El lápiz emite pulsaciones de radio de baja potencia que son detectadas por una trama de alambres bajo el tablero. La circuitería del tablero traduce la posición de la punta del lápiz en coordenadas X e Y que son enviadas al computador. Estos gráficos se utilizan para visualizar la imagen en la pantalla.

El segundo sistema se basa en la utilización de *sprites*, que son áreas de hardware que aceptan imágenes más pequeñas, las cuales pueden ser transferidas al área principal en cualquier momento. En algunos sistemas están dispuestas en profundidad, de manera que las imágenes en los niveles más “próximos” se superponen sobre las que están más lejos. El efecto que se consigue de este modo es muy similar al que los técnicos en películas de dibujos animados logran mediante transparencias.

Como ejemplo podemos imaginar una escena de animación consistente en un hombre andando por la hierba detrás de un árbol y al fondo nubes moviéndose a través de un cielo azul (fig. 14).

En muchos sistemas, cada sprite debe consistir en varias partes de la imagen que tengan el mismo color; de manera que, para conseguir una imagen multicolor, debemos usar más de un sprite. Por tanto, el árbol requiere dos sprites: uno en el plano 0 para el tronco marrón y otro en el plano 1 para las hojas verdes. Estas dos partes del árbol se dibujarán en la pantalla en las posiciones relativas, adecuadas para conseguir la impresión de que se mueven (si decidimos seguir con nuestro hombre paseando) como un todo. El cuerpo del hombre (suponiendo que sea de un solo color) se dibuja en el sprite 2, y tres conjuntos de piernas y brazos en diferentes posiciones en los sprites 3, 4 y 5. Para dar la impresión de movimiento de los miembros, se introducen sucesivamente estos tres sprites en la misma posición en que aparece el cuerpo para que aparezcan unidos a él. Las nubes están dibujadas en el sprite 6, la hierba y el cielo en el plano del fondo que no se mueve.

Estas imágenes pueden dibujarse de diversas maneras: mediante un tablero digitalizador moviendo el cursor por la pantalla con los mandos de control del cursor o con una palanca de mando; escribiendo pequeños programas en lenguajes BASIC o Logo para crear las formas; o combinando formas previamente creadas y almacenadas en un disco o una cinta magnética.

El trabajo de animación resulta así muy sencillo. Se escribe un programa que introduce los diversos sprites en sus posiciones correctas. Empecemos suponiendo que queremos que el árbol y las nubes sean estacionarios y que el hombre ande; se introducen los sprites 0, 1 y 6 en sus posiciones finales; el programa enlaza entonces el cuerpo del hombre, en el sprite 2, en el borde izquierdo de la pantalla, con el primer conjunto de piernas y brazos. Puede ser necesario esperar un tiempo antes de mover al hombre al próximo pixel a la derecha a introducir el segundo conjunto de miembros en esta posición, con objeto de que el movimiento no parezca demasiado rápido. Y lo mismo con el tercer conjunto y de nuevo con el primero. Nuestro hombre “andará” entonces por la pantalla. Cuando llegue al tronco del árbol, parecerá que pasa por detrás de él, ya que los sprites que tienen números más bajos se dibujan sobre los que tienen números más elevados. En cambio, parecerá que

pasa por delante de las nubes, cuyos sprites tienen un número mayor que el que corresponde a cualquiera de los sprites que forman el cuerpo.

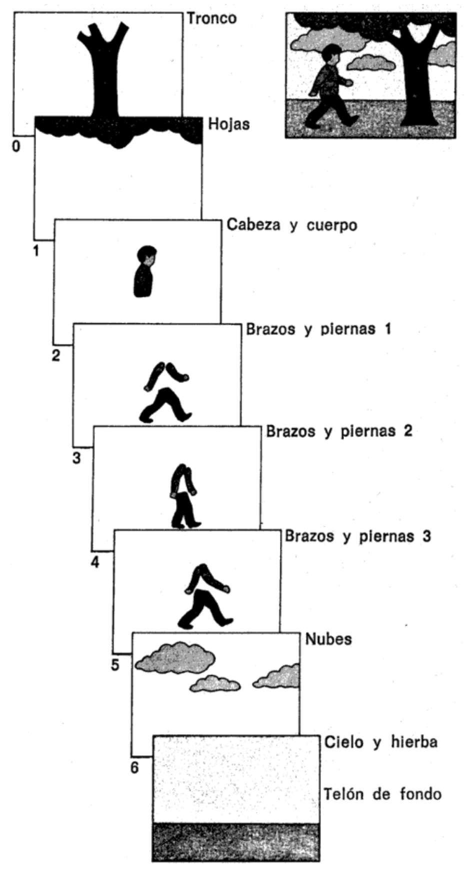


Fig. 14. Modo de utilizar los *sprites* en la animación. Algunos computadores proporcionan varias áreas de imagen que pueden ser representadas en la pantalla en cualquier posición deseada. Las áreas a las que corresponde un número más bajo están representadas sobre las áreas del número más elevado; esto hace posible que, en la secuencia que se presenta, el muchacho aparezca andando por detrás del árbol pero delante de las nubes que se mueven y de la hierba. Las tres posiciones de sus brazos y piernas se introducen sucesivamente en distintos lugares de la pantalla para producir la impresión de movimiento.

Si tuvieran que moverse las nubes (por ejemplo, de izquierda a derecha), el programa de enlace de los sprites debe introducirse en las posiciones apropiadas, comenzando en el lado derecho y avanzando hacia la izquierda.

IMPRESORAS Y TRAZADORES DE GRÁFICOS

A menudo deseamos obtener palabras y figuras en papel en vez de en la pantalla. La máquina que cumple esta función recibe, en buena lógica, el nombre de impresora. En muchos sentidos funciona de forma similar a una máquina de escribir eléctrica. El papel se introduce en la máquina mediante un rodillo de goma cilíndrico y a continuación se escribe. Pero, a diferencia de lo que ocurre en las máquinas de escribir eléctricas, las impresoras raras veces escriben golpeando una cinta con pequeñas cabezas dispuestas al final de brazos metálicos y provistas de tipos. Y es que este mecanismo resulta demasiado frágil para operar a grandes velocidades. Las impresoras utilizan en su lugar otros dos métodos. En el sistema de “margarita” las letras, de metal o plástico, se disponen en círculo alrededor de una rueda. La rueda gira hasta que la letra correcta se encuentra frente a la cinta, siendo golpeada a continuación por un pequeño martillo para que se imprima.

El segundo método es el de la “matriz de puntos”. Este método imita al seguido para obtener letras en la pantalla: utiliza líneas de puntos. Para obtener estos puntos, la cabeza impresora dispone de una fila vertical de rodillos que son golpeados por martillos. Estos a su vez golpean una cinta para así marcar el papel; en algunas máquinas los puntos se escriben mediante chorros de tinta o lásers.

En general, las impresoras de margarita resultan caras, ruidosas y lentas, pero permiten obtener un texto realmente bien presentado. Las máquinas más sofisticadas pueden espaciar las letras entre sí proporcionalmente a su anchura, como puede verse en este texto, donde la ‘i’ es más estrecha que la ‘m’.

Las impresoras de puntos son más rápidas y menos ruidosas, pero, en general, sus impresos son visualmente más toscos. Sin embargo, si los martillos son lo bastante pequeños y están lo suficientemente juntos, el resultado puede ser muy similar al que se obtiene con una máquina eléctrica. Las impresoras de puntos han sido objeto de mejoras enormes en

los últimos años y parece probable que llegará el día en que desplazarán a las de margarita.

La gran ventaja de las impresoras de puntos es que pueden imprimir cualquier forma que su software les indique. Esto significa que es posible pasar de redonda a *cursiva* o a **negrita** en la misma línea de un documento. Y, si es necesario, resulta igualmente fácil pasar de caracteres romanos a caracteres cirílicos, árabes o japoneses. Todos ellos pueden guardarse en ROM en la máquina o enviarse a la misma desde el computador principal.

Las nuevas impresoras de margarita y de puntos ofrecen generalmente modalidad "gráfica", según la cual la cabeza impresora imprime puntos únicos. La cabeza y el rodillo se mueven en pequeños pasos permitiendo que los puntos se superpongan, produciendo así sobre el papel líneas o áreas negras continuas. Los tonos grises se obtienen espaciando los puntos entre sí. Con este sistema es posible obtener imágenes atractivas; pero el software necesario para el control de la impresora es inmenso, ya que se necesitan decenas de miles de puntos y cada uno debe ser individualmente programado de algún modo.

Fig. 15. Letras impresas por una margarita (*arriba*) y una matriz de puntos.

El inconveniente de todas las impresoras es que son mecánicamente complicadas, trabajan sometidas a fuerte tensión y son susceptibles de sufrir graves averías. En los últimos años se han buscado intensamente métodos más simples para realizar marcas sobre el papel y dos soluciones están apareciendo en el mercado. Una de ellas es la impresión mediante chorros de tinta, en la que los puntos se obtienen, no golpeando una cinta con un martillo, sino disparando una gota de tinta sobre el papel. En el más elegante de estos sistemas hasta la fecha, la impresora Olivetti de chorro de tinta *(ink-jet printer)*, se dispara el punto bajo el estímulo de una chispa eléctrica que se produce en el interior de una cápsula de tinta. Este sistema resulta rápido y no es ruidoso y, además, no presenta ningún elemento que pueda deteriorarse por el uso. Cuando la cápsula de tinta se vacía, todo lo que hay que hacer es sustituirla por una nueva.

Otro sistema más caro consiste en escribir el texto con un rayo láser dirigido, imprimir después las marcas mediante algún tipo de proceso xerográfico. Esto resulta mucho más caro, pero, como no hay ningún dispositivo mecánico que toque el papel, puede ser extremadamente rápido. Este sistema es el que tienden a utilizar en la actualidad las compañías de venta directa para producir enormes cantidades de "correo personalizado".

Sin embargo, aunque toda esta tecnología es muy ingeniosa, puede colocar al usuario frente a problemas sorprendentes. Supongamos que se desean imprimir las cartas (en el caso de un escritor, los artículos) proporcionalmente espaciadas con un margen derecho razonable. Es posible obtener el hardware y el software necesarios para realizar la tarea, pero es posible que resulte una agonía intentar que trabajen conjuntamente. El primer problema estriba en que el software de procesamiento de textos calcule el número de palabras que puede situar en una línea. Tiene que separar la última palabra escribiendo un trozo de la misma en la línea siguiente. Después calculará el espacio que le queda para las palabras que quiere situar en la línea. Para que la línea se llene y la última letra de la última palabra se desplace al margen derecho, debe insertar espacios. Si el espacio proporcional es correcto, estos espacios serán fracciones de una pulgada (2,54 cm), que se repartirá entre todas las letras de forma que todas tengan el mismo espacio libre a su alrededor. El computador puede realizar todos estos cálculos y, si la impresora admite un espacio variable, podrán comunicársele los códigos apropiados. Para poder realizar los cálculos existe el problema de que el software debe conocer por anticipado la anchura de cada una de las letras y signos de puntuación en la impresora. Si el usuario ha cometido inocentemente el error de adquirir la impresora y el paquete de procesamiento en dos fuentes distintas, se encontrará con que deberá ser él quien diga al procesador qué espacio necesita en la línea cada letra y esto puede resultar complicado.

En la actualidad, nos encontramos en una fase del desarrollo de los computadores y sus aplicaciones en que existen muchas ideas brillantes, pero muy poca cohesión entre ellas. Los problemas de los usuarios tienen con frecuencia su raíz en la inventiva de los fabricantes. Aunque en principio los computadores pueden hacer cualquier cosa, en la práctica, el que hagan algo tan sólo remotamente útil puede ser tan complicado que no compense el esfuerzo.

Fotocomponedoras de tipos

Los documentos obtenidos por impresoras (que son en realidad sofisticadas máquinas de escribir eléctricas) son una cosa; los obtenidos por impresión propiamente dicha son otra.

Existen varias diferencias. En primer lugar, de la impresión propiamente dicha puede esperarse un estándar mucho más alto de exactitud y equilibrio. Como las letras son más precisas, admiten una disposición de mayor densidad; el texto de un libro, por ejemplo, tiene una composición mucho más densa que el de una carta escrita a máquina. En segundo lugar, el tipógrafo tiene a su disposición una gama mucho más amplia de tipos de letras, espacios entre las letras, y líneas y márgenes de distintos tipos. De hecho son tantas las posibilidades que ofrece la composición tipográfica, que elegir entre ellas es una tarea reservada a los tipógrafos profesionales. Además, son muchos los problemas que resuelven los tipógrafos de los que muy pocos consumidores de letra impresa tenemos alguna idea, aunque, si no dieran con las soluciones idóneas, rápidamente nos daríamos cuenta. Una impresora ordinaria no permite obtener documentos con una composición de tipos correcta; para esto se necesita una “fotocomponedora de tipos”.

Tres son los tipos corrientes de fotocomponedoras, que difieren en el modo de obtener las imágenes en forma de letras que engloban el texto que se les mecanografía. Uno de ellos guarda su archivo de letras y símbolos (de muy distintos tamaños y tipos de imprenta) en negativo fotográfico sobre un disco de vidrio; imprime sobre papel fotográfico moviendo la imagen apropiada del disco al lugar correcto sobre el papel y hace pasar a continuación una luz a través del disco.

El segundo tipo de fotocomponedora dibuja cada letra sobre una pantalla CRT de alta resolución y fotografía el resultado en la posición correcta sobre un papel sensible a la luz. El tercer tipo dibuja cada letra con un láser bajo el control de software.

Lograr que eso funcione no es nada fácil y, para que se obtenga un buen resultado, se necesita una buena dosis de habilidad en el teclado. Un serio problema es la partición de palabras largas que sobrepasan los márgenes derechos en las líneas cortas. Existen pocos programas que realicen esto automáticamente y de forma totalmente satisfactoria.

Recientemente, han comenzado a utilizarse microcomputadores como terminales de fotocomponedoras, de modo que insertan los caracteres de control necesarios en el texto que se desea someter al procesamiento. Esto posibilita que los documentos obtenidos en microcomputadores vayan directamente a la fotocomposición sin necesidad de ser "retecleados"; ello supone un gran ahorro de trabajo, aunque la innovación no goza lógicamente de popularidad en los sindicatos de artes gráficas.

Dispositivos trazadores de gráficos

En la página 38 vimos que con una impresora se obtienen imágenes de aspecto bastante tosco. Una solución más sofisticada para la obtención de gráficos es la construcción de una nueva máquina completamente separada, un dispositivo trazador de gráficos *(plotter),* que se sirve de un lápiz y dibuja de forma similar a como lo hace una persona.

Esencialmente, un plotter consiste en una pluma accionada por dos motores que la mueven en pequeños pasos horizontales y verticales. Moviendo la pluma el número apropiado de pasos cada vez, horizontal y verticalmente, se consigue desplazarla en la dirección deseada.

Si se mueve la pluma sin pasos verticales, dibujará una línea horizontal; cuando se mueve sin pasos horizontales, dibujará una línea vertical. Si el número de pasos horizontales y verticales es el mismo, dibujará una línea con 45 grados de inclinación. Si la longitud de cada paso es de aproximadamente una décima parte del grueso de la línea que traza la pluma, los pasos resultarán invisibles y el efecto visual será el de un dibujo continuo.

Cuanto más caro sea un plotter, más se aproximará a este ideal, pero los realmente buenos son muy caros. En la actualidad, presentan una gama de plumas de distintos colores y resulta realmente divertido ver cómo se para y coge una pluma verde del bastidor, escribe algo en verde, se detiene nuevamente para buscar una pluma roja, etc.

Sin embargo, el coste no es tan importante, ya que estas máquinas se utilizan para diseños de ingeniería en proyectos de gran envergadura y presupuesto. Es fácil coordinar el trabajo de muchos ingenieros si se conservan las instrucciones para el plotter en una base de datos central;

cuando se utiliza un plotter para obtener los dibujos finales, se ahorra el trabajo de muchos delineantes.

El ingeniero que diseña un plotter se enfrenta a problemas bastante complejos. Si no fuese inconveniente que la máquina tardara un año en hacer un dibujo, las cosas resultarían mucho más fáciles, pero se trata de que haga el trabajo con la misma rapidez que un delineante. El diseñador tendrá, por tanto, que solucionar problemas tales como los de exceso de trabajo de los motores que impulsan los pasos, que harán que la pluma se balancee en torno a su nueva posición a menos que espere una o dos milésimas de segundo antes de escribir. El diseñador tendrá que prever la acumulación de suciedad en los engranajes y cuerdas que mueven las plumas, para que éstas no sean conducidas a posiciones distintas según de donde provenga la suciedad.

Si el plotter debe trabajar con rapidez, se acelera la cabeza drásticamente. Los plotters de platina más grande que se han construido utilizan cables del grosor de un dedo, capaces de soportar las cargas mecánicas necesarias para lograr tal aceleración, y, mientras trabajan, deben estar cubiertos de una tapa de vidrio que evitará que las manos resulten dañadas por el plotter en movimiento.

Muchas de las marcas que aparecen en un diseño de ingeniería son letras o figuras estandarizadas tales como círculos, cuadrados y elipses. Por tanto, el software del plotter, que controla la cabeza, tiene algunas de las funciones de una impresora. Para escribir el nombre de una pieza no es necesario guiar a la pluma alrededor de las letras, como si se tratase de elementos de maquinaria, es suficiente escribir en el teclado: PRINT “Diente de la corona dentada 1/4” o “TRAZAR CÍRCULO 2,6; 3,56; 6,1”, siendo los números las coordenadas del centro y el radio. Un plotter por sí solo tiene la misma utilidad que una impresora aislada: necesita que lo guíe un software, que realizará el mismo tipo de operaciones que el software de procesamiento de textos hace con una impresora. Sin embargo, el software para crear y manejar imágenes tridimensionales de proyectos de ingeniería no es sencillo ni sus demandas al hardware son triviales.

MEMORIA MAGNÉTICA

En las páginas 20-22 hablamos de la memoria electrónica que se emplea dentro del propio computador y que está conectada directamente al procesador para permitir el rápido acceso a la misma. Pero como esta memoria resulta bastante cara incluyendo todos los chips para refrescarla y controlar sus buses, necesitamos disponer de otra memoria alternativa que, aunque sea más lenta, resulte más barata. Estos dos tipos de memoria pueden compararse respectivamente con los papeles que se tienen sobre la mesa de trabajo, entre los que es posible encontrar rápidamente el que se necesita, y la gran masa de documentos que se guardan en el archivador, donde lleva mucho más tiempo localizar el que se busca.

Todo lo que se necesita para tener una memoria es algún tipo de efecto físico que pueda ser provocado eléctricamente con el fin de que el computador pueda escribirlo, y que a la vez provoque un efecto eléctrico tal que el computador pueda leerlo. En principio no importa de qué efecto se trate, y a lo largo de los años se han usado métodos realmente curiosos. Una de las primeras memorias de computador, la construida al final de los años cuarenta para el computador Mark 1 en Manchester, Inglaterra, utilizaba pequeñas masas de carga escritas sobre un tubo de rayos catódicos. Otro método muy simple y seguro, aunque lento, es utilizar agujeros perforados sobre una cinta de papel. Pueden hacerse con un punzón activado eléctricamente y leerse con haces de luz o pequeños contactos eléctricos. Algunas instalaciones de computadores de tamaño considerable todavía utilizan cintas de papel.

Como el inventor de una memoria para computador barata se enriquecería más allá de lo imaginable, se han explorado un número considerable de posibles tecnologías. Es interesante constatar que, de todos los métodos que se han propuesto, el que ha superado la prueba del tiempo y se emplea en la actualidad casi universalmente es el del registro magnético. Consiste en revestir una superficie apropiada con una emulsión magnética y magnetizar después pequeñas áreas de la misma en una o dos direcciones para que registre un ‘1’ o un ‘0’. (En la práctica resulta algo más complicado: lo que registra ‘1’ es un *cambio* de norte a sur del eje magnético; el ‘0’ se registra por un cambio de sur a norte.) Lo mejor del registro magnético es que se automantiene. Los minúsculos imanes que forman un área apuntan todos arriba o abajo según registren 1 o 0, y si

uno de ellos se sale de la línea, los otros lo empujan a que vuelva otra vez a su sitio.

Los sumisos imanes están todos situados de manera que el polo norte de uno se encuentra próximo al polo sur del otro, que es exactamente su posición idónea. Cualquier error se traduce en un imán que de algún modo girará sobre sí mismo de forma que su polo norte se aproximará a los otros polos norte y su polo sur a los otros polos sur. Cualquiera que haya jugado con un par de imanes sabe que esto genera una fuerza de repulsión, que obligará pronto al imán transgresor a girar hasta volver a la posición correcta. Esta característica significa que los datos escritos por medios magnéticos pueden guardarse durante un tiempo muy largo (muy largo si se mide con el estándar de tiempo que se emplea en informática, que es de millonésimas de segundo) sin que sufran alteraciones. Por regla general, se calcula que los datos deben regrabarse cada dos años, de otro modo se ven afectados por la fatal y misteriosa "putrefacción de bytes".

Hay muy pocos procesos físicos que tengan esta capacidad de automantenimiento. Por esta razón, los ordenadores utilizan memorias magnéticas. Otras ideas se perfilan en el horizonte, pero nada parece que pueda sustituir los registros magnéticos.

Una vez se dispone de un sistema para realizar marcas indelebles (muy similares a las de la tinta en papel), se necesita un método para acceder fácil y rápidamente a cualquier marca determinada. En una grabadora de cinta para voz o música, la emulsión magnética reviste una cinta de plástico delgada y flexible, que es arrastrada con velocidad constante para que pase por una cabeza de lectura/escritura. Como la música tiene carácter serial y al escucharla no se desea saltar de un punto de la grabación a otros ni situarse directamente en mitad de una pieza, este sistema funciona bastante bien.

A falta de una técnica más idónea, durante muchos años se han utilizado cintas magnéticas para almacenar datos, hasta el punto de que máquinas provistas de grandes rollos de cinta se han convertido en el símbolo visual estandarizado del "computador" en las películas y la televisión, aunque hoy en día las cintas magnéticas como método de almacenamiento están ya en desuso. Lo que hace a las cintas adecuadas para la música —su naturaleza serial—, las hace inadecuadas para servir de memoria, como sabe por propia experiencia cualquiera que posea un microcompu-

tador y utilice una cassette. El problema radica en que, con las cintas, resulta obligatorio leer los archivos desde el principio hasta el final.

Para mejorar la situación, alguien tuvo la brillante idea de combinar las mejores características de las cintas magnéticas y de los discos de gramófono: se extiende una emulsión magnética sobre la superficie de un disco que gira, mientras la cabeza que lee y escribe las pequeñas áreas magnéticas se mueve hacia delante y hacia atrás desde el centro al borde. Con este método se reduce enormemente el tiempo necesario para llegar a cualquier punto del disco, aunque precisa de un complicado sistema de engranajes.

Los datos se escriben en “sectores”, que son partes del círculo, y en “pistas”, que son círculos de distinto radio. Las pistas no están sobre una espirar como en los discos de gramófono. Un pequeño agujero en el disco, cerca de su centro, deja pasar un rayo de luz a cada revolución, de manera que la circuitería en la unidad de gobierno del disco puede averiguar dónde se encuentra la cabeza en relación con todos los sectores. En este sistema conocido como *soft sector*, los sectores son seleccionados electrónicamente por una señal reguladora del agujero indicador. Algunas máquinas utilizan discos *hard sector*, que tienen un agujero en el disco para cada sector. De este modo resulta muy sencillo leer los tres conjuntos de datos a los que tan difícil nos resultaba acceder en la cinta. Sólo tenemos que mover la cabeza hacia delante y hacia atrás hasta situarla en la pista correcta y esperar que llegue el sector correcto. El tiempo medio que se tarda en mover la cabeza desde una pista cualquiera a otra se denomina *seek time* (tiempo de búsqueda), y el tiempo medio de espera hasta que llega al sector que se desea se conoce como *latency* (latencia).

En la práctica, un disco flexible con una unidad de gobierno bien programada (véanse pp. 65-72) debería tardar entre 1/3 y 1/5 de segundo en encontrar cualquier cosa determinada; un disco duro debería ser por lo menos dos veces más rápido. Hoy día, prácticamente todos los microcomputadores de precio medio tienen una unidad de discos, y ello se está extendiendo a los computadores personales. La mayoría de estas unidades son para *floppies*, en los que el material de grabación se encuentra en un disco de plástico fino y flexible contenido en un sobre cuadrado. Los tamaños corrientes son los de 8, 5¼, 3 y 3¼ pulgadas (20,30; 14,30; 7,62 y 8,25 centímetros, respectivamente).

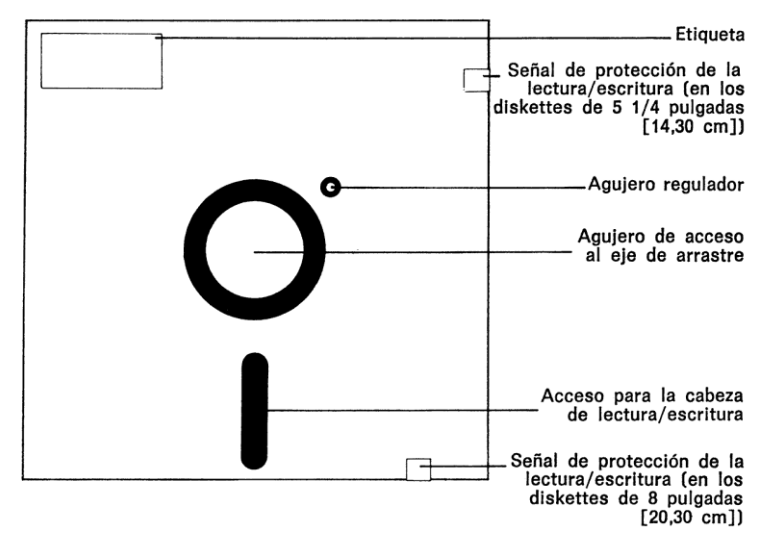


Fig. 16. Un disco flexible. El disco propiamente dicho —un círculo de plástico flexible, revestido de material magnético— está contenido en una funda de plástico cuadrada, de la que no puede ser extraído. Nunca debe tocarse la superficie del disco; tampoco hay que doblarla, calentarla o fijarla con grapas ni escribir sobre él con un bolígrafo. Evite dejarlo bajo el teléfono, ya que el campo magnético del timbre podría estropearlo. Para utilizarlo, se introduce en el arrastre de manera que la etiqueta mire hacia la puerta.

Lo que encarece una unidad de disco es que requiere una ingeniería de gran precisión que garantice que se sitúe con exactitud la minúscula cabeza de lectura/escritura sobre la pista que se desee. En varias ocasiones se han producido intentos de diseñar dispositivos que combinen las ventajas de precio de las unidades de cinta con la velocidad de acceso de los discos. Uno de estos dispositivos fue llamado *The Siringa Floppy*, pero no tuvo mucha aceptación. En el momento de escribir este libro, sir Clive Sinclair está a punto de introducir en el mercado, desde Inglaterra, un nuevo dispositivo llamado *Micro-Drive*[3]. Tiene el aspecto de un disco,

[3] Al realizarse la edición en lengua castellana, este dispositivo ya ha aparecido en el mercado, bajo el soporte del computador Sinclair Q-L.

pero posee un mecanismo que enrolla y desenrolla la cabeza en espiral desde el borde al centro y al revés. Esto permitirá abaratar el hardware, pero será lento.

Los floppies pueden ser de una o dos caras; aunque un floppy de dos caras no tiene sentido si la unidad de disco no posee dos conjuntos de cabezas. Las unidades de disco pueden ser de densidad simple, doble o cuádruple; esta división hace referencia al número de pistas que ponen en el floppy o que leen de él. Desde determinado punto de vista la cuádruple es la mejor porque almacena cuatro veces más datos; pero desde otro punto de vista es la peor, porque la cabeza debe situarse con cuatro veces más exactitud para que lea los datos correctamente. Ocurre con frecuencia que un disco de densidad cuádruple escrito en una máquina determinada no funciona en otra máquina idéntica. Para la transferencia de datos entre máquinas, los discos de densidad simple son los mejores. Pero una vez que se ha establecido en la máquina un programa o un archivo de datos, puede usarse con seguridad la opción de la densidad cuádruple, ya que los errores de la cabeza que se produzcan al leer serán los mismos que se produjeron al escribir y la cosa funcionará.

La unidad de floppies está conectada al tablero principal del computador por un cable de muchas vías. Generalmente escribe y lee de la memoria a través de un chip de acceso directo a memoria *(direct memory access* o DMA): una especie de procesador especializado, que no hace nada más que leer y escribir en la memoria paralelamente al procesador principal. La transferencia de datos entre discos se realiza a unos 250 KB/s.

Aunque los discos duros se están popularizando a medida que disminuyen de precio, todavía son muchas las ventajas de los floppies o discos flexibles: son muy baratos; pueden almacenar gran cantidad de datos (hasta 500 KB por cara); permiten hacer una copia de los programas y datos de mayor importancia y guardarlos fácilmente en lugar seguro; se remiten por correo sin más trámite que poner el disco en un sobre. Tienen, sin embargo, el inconveniente de que sufren mucho desgaste y se deterioran hasta quedar inservibles en plazos de tiempo imprevisibles. Para su tranquilidad, haga siempre segundas copias de sus archivos importantes.

Lo más irritante de los minifloppies es que un disco escrito en una máquina probablemente no podrá leerse en otra, porque cada fabricante

tiene su sistema particular de escribir datos en el disco. Aunque todo el mundo conoce las dificultades que esto comporta, los nuevos discos de 3¼ pulgadas (8,25 cm) parece que no han logrado superar el problema de los minifloppies.

Existe un estándar de floppy de 8 pulgadas (20,30 cm) un disco de simple cara y de simple densidad que funcionará bastante bien en cualquier máquina; se le conoce como SSSD (*single-sided single-density disk).* Esta cualidad hace del SSSD de 8 pulgadas (20,30 cm) un soporte idóneo para la distribución de software. Para obtener software de una máquina que lee discos de 8 pulgadas para una que los escribe en el formato de 5⅕ pulgadas (14,30 cm), hay que ponerlas físicamente en contacto y trasvasar los datos a través de un interface RS 232 (véanse pp. 38-40).

Las buenas noticias en los últimos años nos las han dado los fabricantes de discos duros. La explicación es que cuanto menores puedan hacerse los puntos que almacenan datos, mayor cantidad de éstos podrán almacenarse en una determinada área del disco. Esto da al usuario final mayor capacidad sin incrementar el tiempo global de respuesta, porque la cabeza se mueve exactamente a las mismas distancias.

La forma de hacer estas áreas pequeñas es situar la cabeza de registro magnético más próxima a la superficie del disco. Esto significa que es necesario sustituir los antiguos floppies flexibles por discos rígidos, lisos y duros. Para evitar graves deterioros hay que impedir el contacto con las impurezas que flotan en el aire (polvo, pelos e incluso partículas de humo de cigarrillos); con ese fin se dispone el disco dentro de un espacio protegido. Por otra parte, la distancia entre la cabeza y el disco es tan pequeña (alrededor de 18 millonésimas de pulgada), que no puede ser mantenida con suficiente exactitud por medios mecánicos. La cabeza realmente vuela en el viento que levanta el disco al girar; un pequeño muelle impide que se despegue por la presión del aire. Cuando el disco para, la cabeza “aterriza” en una pista especial en la que no existen datos. El conjunto constituye una maravilla de imaginación y puede almacenar 50 MB en una caja del tamaño de una unidad de floppies de 5¼ pulgadas (14,30 cm). Un buen mecanógrafo puede teclear unas 1.000 palabras por hora como promedio, de manera que para escribir 30 MB de información necesitaría 623 días (más de dos años).

Pero la búsqueda de maneras de incrementar la capacidad de almacenamiento continúa. El sistema en auge comporta el paso del registro horizontal, en el que las pequeñas áreas magnéticas se hallan sobre la superficie del disco, al registro vertical, en el que van de un lado del disco al otro. Se cree que de este modo es posible incrementar la densidad de registro cerca de 40 veces; de manera que un minifloppy podría guardar 20 MB de datos, y un disco duro 1.200 MB, lo que equivale a los datos escritos a máquina en ochenta años.

Burbujas magnéticas

Parece absurdo que el almacenamiento de datos dependa de motores y discos que giran, cuando parece que lo lógico sería que existiera algún procedimiento más elegante y sencillo. El problema con los discos no reside en su escasa elegancia, sino en que las partes móviles fallan con facilidad. La cabeza, en particular, se ensucia o deteriora fácilmente, con consecuencias imprevisibles para los datos que se encuentren almacenados en los discos. Una aparente solución, de la que se esperaba mucho a finales de la década de los setenta, es el almacenamiento en burbujas magnéticas. La idea era codificar los datos en pequeños “dominios” magnéticos, pequeños volúmenes en los que la información magnética codificada se automantuviese en su sitio en la forma usual, y, en vez de mover estos dominios para que pasen por una cabeza de lectura/escritura, dejarlos flotar en el material magnético.

El resultado fue la “memoria de burbujas”, en la que pequeñas “burbujas” magnéticas se empujaban a través de una fina capa de material magnético mediante campos magnéticos externos. Ingeniosamente, se organizaba el material de diversas maneras para que las burbujas se comportasen de forma adecuada. Se codificaba un conjunto de datos magnetizando una serie de burbujas en sentido norte-sur (para representar un 1 binario) o en sentido sur-norte (para representar un 0). Con esta serie de burbujas se formaba un lazo, que podía enrollarse sobre sí mismo para que llenara todo el material magnético disponible; las burbujas circulaban por el lazo pasando sucesivamente por una cabeza de lectura/escritura construida en el material magnético. Era esencialmente una cinta sin fin; pero en este caso la cinta permanecía estática y eran los

datos los que se movían. Resultó que era posible mover los datos con bastante velocidad por el material, de manera que se salvaba la desventaja principal del almacenamiento en cinta: la lentitud. Sin embargo, el proceso exigía disponer de bobinas magnéticas bastante sofisticadas, difíciles de fabricar.

Sin duda, si la memoria de burbujas se hubiera implantado comercialmente, se habría abaratado. No obstante, como el volumen de ventas de discos convencionales es muy grande, éstos resultan ya muy baratos en la actualidad; las burbujas no están en condiciones de competir. Se utilizan principalmente para computadores en aviones o barcos de guerra, donde los discos no ofrecen garantías suficientes por su sensibilidad al polvo o a cualquier alteración, que en este caso podría producirse fácilmente a causa del fuego enemigo, y donde la importancia del mayor coste es sólo relativa.

ARCHIVOS Y SISTEMAS OPERATIVOS

Un disco ofrece, por sí mismo, espacio para almacenar en bruto. Es como un archivador vacío, de poca utilidad sin cajones ni fichas. Las fichas deben estar etiquetadas y clasificadas de alguna manera para que se puedan encontrar los documentos que se han guardado. En un computador, este trabajo lo realiza el “sistema operativo”, programa que dirige todas las tareas de mantenimiento *(housekeeping)*. En muchos sentidos, por lo que se refiere al usuario, este sistema es el propio computador.

Al nivel más bajo, un sistema operativo realiza una serie de tareas vulgares pero esenciales. Cuando se escribe en BASIC:

```
10 INPUT «Entrar el próximo número»; N
```

El BASIC (que es un programa para entender lo que se quiere decir con este tipo de frase) transmite la serie “Entrar el próximo número” al sistema operativo con una orden para que la imprima en la pantalla. Entonces, la orden espera una entrada del teclado y aguarda a que el sistema operativo se la proporcione. Cuando el usuario ha escrito un número y pulsado la tecla RETURN (RETORNO) el número se transmite de nuevo al BASIC.

Naturalmente, el código para hacer estas operaciones podría estar escrito en BASIC. La razón por la que no lo está reside en el hecho de que quizá quieran desarrollarse otros lenguajes o programas escritos en el código de máquina. Estos lenguajes o programas conviene que sean capaces de imprimir mensajes en la pantalla y que acepten entradas provenientes del teclado, de modo que, para ahorrar esfuerzos, tiene sentido escribir las rutinas una vez y dejar que todo el mundo las utilice. Además, es factible la estandarización del modo como circula la información desde (y hacia) el sistema operativo, mientras se garantice que los paquetes de software estandarizado imprimen en la pantalla y obtienen textos del teclado.

Un sistema operativo manipulará también la impresora, enviándole textos a la velocidad adecuada, al tiempo que reconoce sus señales de *hand-shaking* (véanse pp. 38-40).

También realizará la delicada tarea de almacenar información en el disco y recuperarla. El problema en este caso consiste en utilizar de la mejor manera el espacio disponible. Esto resulta bastante fácil si se empieza con un disco vacío. Se escribe el primer archivo y a continuación el segundo y luego el tercero. Cuando se llega al centro, se ha llenado el disco y se detiene la operación. Sin embargo, mucho antes de que esto ocurra se habrá recuperado, casi con toda seguridad, el primer archivo, se habrán hecho algunos cambios, se habrá borrado la primera versión y grabado la nueva. Es probable que la nueva versión no ocupe exactamente el mismo espacio; será demasiado grande o demasiado pequeña. Muy pronto el disco estará en un terrible desorden con gran número de espacios disponibles cuyo tamaño no será suficiente para admitir archivos completos.

El esquema estandarizado consiste en dividir el disco en pequeños trozos de almacenamiento, llamados normalmente *records* (que no deben confundirse con los “Records” de una base de datos; véanse pp. 150-155). El sistema operativo mantiene un directorio, escrito normalmente en las dos pistas más exteriores del disco, para indicar qué trozos se están utilizando y qué archivo los utiliza. Así, el archivo 1 podría estar escrito en los records 34, 35, 36, 47, 53, 96, 97, 98, 99, 100, el archivo 2 en 2, 3, 4, 5, 6, 7, 26, 29, 39, 126, el archivo 3 en otros, mientras que muchos otros records están en blanco. Cuando se borra un archivo, sus records quedan señalados en el directorio como disponibles; cuando se escribe un

archivo de n records de longitud, se coloca en los primeros n records en blanco en el directorio. Este sencillo esquema permite usar eficazmente el disco aunque a costa de que se mantenga un directorio y de que la cabeza tenga que hacer muchos saltos si el disco se ha utilizado a menudo.

El usuario no tiene necesidad de saber nada de esto. Por lo que a él concierne, se limita a pedir al sistema operativo que escriba y lea archivos, y esto es lo que hace. Cómo hacerlo es su verdadero trabajo.

Antes de continuar, quizá sea interesante echar una mirada al importante concepto de "archivo", ya que muchas de las operaciones del computador giran a su alrededor. Un archivo es simplemente una larga secuencia de bytes (¿qué otra cosa podría ser?) escrita en un disco. Tiene un nombre, un principio y un final. El nombre está en el directorio con un número que indica el record en el que empieza el archivo. Para indicar exactamente dónde termina el archivo, hay un marcador de fin de archivo *(end-of-file;* EOF). Cada vez que se lee un archivo, el sistema operativo comprueba cada uno de los bytes que provienen del disco, buscando la señal EOF. Cuando encuentra uno, detiene la lectura. Si, debido a un accidente, encontramos un EOF en medio de un archivo, entonces no cabe duda de que tendremos problemas.

Los bytes de un archivo pueden considerarse de dos maneras: como un tipo particular de datos o como un programa. Los datos serían quizás un texto de archivo que puede ser traducido a caracteres alfabéticos o a la inversa mediante un paquete de tratamiento de textos; o bien una serie de bytes que representen números (la salida de una nómina o un paquete de control de stocks) mezclados con algunos bytes que representen texto e incluso bytes que representen coordenadas de los puntos de un dibujo realizado en la pantalla del computador. A menudo, no puede saberse lo que hay en el archivo sólo mirándolo, sino que debe ser leído por el programa adecuado para que el conjunto adquiera sentido.

Si el archivo es un programa, sus bytes serán interpretados como instrucciones del código en lenguaje máquina (véanse pp. 104-120) y direcciones de memoria. Casi con absoluta seguridad habrán algunos datos mezclados en el programa, pero los bits de programa del archivo reconocerán cuándo se trata de un dato y cuándo no. El sistema operativo realiza la "interpretación" guiado por el nombre del archivo. Según la tradición, los nombres que se le dan al archivo deben constar de dos partes:

un nombre y una extensión, para identificar el archivo e identificar al sistema operativo de qué tipo es.

La diferencia crucial entre los dos tipos de archivo radica en que, cuando se da al sistema operativo el nombre del archivo de un programa, sabe lo que tiene que hacer con él, o sea: leerlo en el disco, cargarlo en la memoria con su principio en el lugar donde se inician los programas y hacerlo funcionar. A todos los demás archivos sólo se accede mediante programas, ya que el sistema operativo no sabe por sí mismo qué hacer con ellos.

Un buen sistema operativo hace mucho más que esto. Debe tener programas sencillos (llamados normalmente "programas de utilidad" para distinguirlos de los programas que hacen algo útil en el mundo real) que informen del espacio que queda en el disco o sobre el tamaño de los archivos, que den nuevos nombres a los archivos y los copien de un disco a otro. También es importante que permita al usuario determinar los tipos y velocidades de impresión. Si el sistema permite trabajar a varios usuarios a la vez (véase p. 210), es muy útil que prevea asimismo la posibilidad de que un usuario interfiera con el archivo de otro. Otra prestación importante es el control del modo como los usuarios escriben y leen los archivos compartidos, de manera que uno no trate de leer un archivo en el que se está escribiendo. Por último, el sistema operativo debe permitir que un usuario envíe mensajes a otro o al mundo exterior.

En el mundo real de los microcomputadores existen varios grupos de sistemas operativos cuyos fabricantes heredaron el esquema de los mainframe, según el cual cada fabricante proporciona automáticamente a sus clientes su propio sistema operativo. Y esto es así, en parte, para impedir que los clientes compren en otro sitio la parte más provechosa del paquete de programas: el software.

Los sistemas característicos de cada fabricante, aunque proporcionan en cada máquina individual las prestaciones que todo sistema operativo debería ofrecer, no agotan las ventajas inherentes a la idea básica: un sistema operativo común, en cambio, logra que las máquinas de diseño distinto parezcan iguales.

Esto lo descubrieron a mediados de la década de los setenta casi por azar las personas que utilizaban microcomputadores. Ello ocurrió cuando Gary Kidall escribió un software para obtener datos o para introducirlos

en el disco de un microcomputador. A este software lo denominó *Control Printer/Monitor* (Control de pantalla e impresora).

El resultado fue el CP/M, que obtuvo un enorme éxito y fue utilizado por docenas y luego por cientos de fabricantes de microcomputadores 8080 y Z80. En el curso del tiempo, la historia hizo que el CP/M se convirtiese en *Control Program for Microcomputers* (Programa de control para microcomputadores).

Un sistema operativo común, como el CP/M, permite que el mercado de software obtenga un número mucho mayor de clientes de los que conseguiría en otro caso.

Utilizar un sistema específico para una máquina —incluso si fuera más eficaz que el CP/M— sería algo parecido a imprimir un libro en finlandés en lugar de en castellano, porque la lógica del lenguaje es más adecuada a su tema. Por desgracia, hay muchos menos lectores que saben finlandés que castellano. El desarrollo de un mercado masivo para el software sería imposible sin sistemas operativos comunes.

Por desgracia, ser propietario de un sistema operativo ampliamente utilizado resulta tan provechoso que más de uno intenta introducirse en el mercado. La consecuencia que sigue a ello es la fragmentación del mismo.

La fragmentación del mercado de sistemas operativos no es positiva por varias razones. En primer lugar, un sistema operativo general crea un amplio mercado, que a su vez atrae muchos productos de software. La competencia fuerza el abaratamiento de los productos y mejora sus cualidades, lo que beneficia al usuario. En segundo lugar, si varios fabricantes proporcionan el mismo sistema operativo, ninguno de ellos podrá dominar el mercado en exclusiva. Un fabricante decidió hace pocos años que no le gustaba que otros vendieran software a sus clientes. Para impedirlo, cambió su sistema operativo. El resultado fue que dicho fabricante sufrió tanto como cualquier otro, porque los usuarios reaccionaron en contra de esta intimidación. Pero si, por ejemplo, una docena de fabricantes hubiesen fabricado máquinas con este sistema operativo y utilizaran el mismo software, entonces ninguno de ellos podría haber actuado de esta manera sin perjudicarse a sí mismo muchísimo más que a cualquier otro.

Considerado desde este punto de vista, en la actualidad hay varios sistemas en el mercado de microcomputadores. El mayor de ellos es el CP/M (y sus derivados y mejoras tales como MS-DOS). Este sistema ha

crecido al mismo ritmo que el mercado y es adecuado y sencillo. Según los estándares propios de los grandes computadores es tan simple como un juego de niños, pero realiza muy bien todo lo que el usuario desea y parece que se está adaptando a las pequeñas redes de microcomputadores, denominándose en este caso CP/Net. Existen varias redes similares de sistemas operativos, CP/M *look-alike,* tales como Turbodos, MacNos, MMost. Actualmente se han unido varios fabricantes para lanzar al mercado uno nuevo llamado MSX, aunque inicialmente sólo es utilizado en pequeños computadores.

El rival para CP/M y MS-DOS en sus varios formatos es UNIX, un sistema operativo concebido inicialmente para minicomputadores multiusuario, hace diez años aproximadamente, en los laboratorios Bell en Estados Unidos. Si CP/M es demasiado simple, UNIX parece demasiado complejo. En su forma actual es una herramienta para el programador profesional que permite a los usuarios poner en marcha procesos completos con diversos programas mediante la simple pulsación de una tecla. La salida de un programa puede ser "conducida" a la entrada de otro. Tiene un mecanismo, llamado *shell* (concha), que permite a cualquier programa ejecutar otro como si fuera el sistema operativo.

UNIX es muy apropiado para los programadores profesionales; no obstante, en su forma completa crearía a los usuarios de computadores ordinarios demasiados quebraderos de cabeza. Pero esto no es excesivamente importante, ya que al programador le es muy fácil adaptarlo al diseño que adquirirá el usuario final. En realidad, cualquiera que sea el sistema elegido podrá lograrse que se comporte como CP/M si los usuarios lo desean, de manera que éstos puedan ejecutar software CP/M con él.

Smalltalk

Hasta ahora hemos examinado el funcionamiento del sistema operativo al nivel más bajo de la máquina: tareas internas entre los discos, teclado, pantalla e impresora. El sistema operativo también tiene responsabilidad al nivel más alto, porque sólo a través de él el usuario puede ejecutar programas.

Una solución es crear nuevos sistemas operativos que carguen automáticamente series completas de programas, alimentándolos, si es necesario, con órdenes que, en otro caso, el usuario debería leer en el manual y escribir en el teclado. Otro planteamiento denominado *Smalltalk*, desarrollado por Xerox en su centro de investigación de Palo Alto y seguido por Apple con el Lisa, consiste en rechazar el listado alfabético de los programas y los archivos de datos, y proporcionar a los usuarios algo con lo que se sientan más cómodos. La solución Smalltalk consiste en presentar a los usuarios dibujos de cosas a las que puedan dar algún significado: el dibujo de una carpeta significa un archivo de datos; una impresora significa la impresora; y un cubo de basura es donde se ponen las cosas de las que uno quiere deshacerse.

Para que estos dibujos operen, se dispone de un cursor controlado por un "ratón" (véase p. 38). Al mover el ratón por la superficie de la mesa de despacho, el cursor se mueve por la pantalla. Cuando el ratón está sobre el dibujo que se quiere ejecutar, se aprieta un botón. Si se quiere imprimir un archivo concreto, se dirige el cursor hasta él, se aprieta el botón y, a continuación, se conduce el cursor al dibujo de la impresora. Se aprieta de nuevo el botón y se manda a imprimir el archivo. Si se quiere editar otra ficha, se coloca el cursor sobre el dibujo del editor para cargarlo y luego sobre el archivo que se quiere cargar y representar en la pantalla.

Naturalmente, todo este proceso consume mucha capacidad de memoria. La pantalla debe tener una gran resolución para presentar imágenes con suficiente detalle. Además, la aplicación Lisa permite obtener en la pantalla varias páginas seguidas de documentos, una encima de la otra, como si fueran documentos apilados ordenadamente sobre una mesa de despacho.

Se pueden extraer fragmentos de un documento e introducirlos en otro; para ello se coge, por ejemplo, una parte de una "hoja de contabilidad" también llamada "hoja electrónica" *(spread sheet;* véase p. 161) y se pega en un informe que está siendo preparado por el procesador de textos. Hay un paquete de programas de dibujo que permite al usuario dibujar croquis, guardarlos, recuperarlos e incorporarlos en los documentos. También existen, entre otras cosas, un paquete de programas de dibujo de gráficos y un gestor rudimentario de la base de datos.

Todo esto requiere fuertes prestaciones de hardware. Lisa tiene un procesador 68000 —el más potente entre los de las máquinas de 16 bits— y 2 MB de RAM, lo que hace a su vez que sea una máquina cara. A juzgar por la expectación que genera cualquier demostración de Lisa, este computador atrae enormemente a los usuarios ingenuos. El tiempo dirá si el público seguirá dejándose engañar alegremente pagando lo que siempre será un alto precio extra, por el hardware necesario para imitar documentos, en lugar de pasar unas cuantas horas aprendiendo el modo de hacer las cosas de una forma más económica. Todavía es más preocupante (aunque las "metáforas de documentos" en lugar de ideas de computación hacen que resulte mucho más fácil vender computadores a los usuarios inexpertos) el hecho probable de que a la larga ejerzan una influencia esterilizadora. Tal como he tratado y seguido tratando de demostrar en este libro, hay muchos aspectos de la informática que no tiene paralelo en el mundo de la informática escrita. Tarde o temprano, las personas que quieran ser informatizadas se rendirán ante la evidencia de que informatizarse es bien distinto que trabajar con papeles.

SOFTWARE DOMÉSTICO

El boom de los chips baratos ha producido un boom aún más espectacular en los computadores baratos y pequeños: computadores "domésticos" o "personales", como parece que finalmente el mercado ha decidido denominarlos.

En el momento en que escribimos este libro estas máquinas tendían a presentar un hardware bastante simple. Sus procesadores eran Z80 o 6502, y solían tener pantallas de 40 × 20 o aún menores en algunos casos. En ocasiones sólo representaban en la pantalla mayúsculas y algunos gráficos bastante elementales. Tenían color, naturalmente, mientras usted fuese propietario de un televisor en color para conectarlo a la máquina y nadie en su familia quisiera ver "la película". A menudo tenían menos de 64 K de RAM; entre 16 K y 48 K eran las cantidades más comunes. Desde el punto de vista de la microinformática comercial, lo menos satisfactorio es que carecían de discos. Su soporte de almacenamiento era la cassette. Si ésta no funcionaba correctamente, no se obtenía copia alguna de lo almacenado. Además, incluso en el caso de que funcionara correctamente, sólo podía accederse a su fichero de datos según una determina-

da secuencia (véase p. 28). Un microcomputador es realmente útil sólo cuando puede almacenar en cualquier orden, mucho más de 20 K de datos en la memoria (tras haber cargado un lenguaje).

Estas limitaciones implicaban que los usuarios de computadores domésticos estaban obligados a utilizar programas que se ejecutasen completamente en la memoria. Sin embargo, de la noche a la mañana surgió un asombroso mercado que suministraba software para máquinas pequeñas. Consistía principalmente en juegos. Después de los juegos, y a gran distancia en popularidad, estaban los programas educativos que trataban de explicar, en un marco parecido al de los juegos, los elementos de álgebra, física o idiomas.

Durante cierto tiempo pareció que la enseñanza sería una de las tareas más importantes de los microcomputadores. Se argumentaba que del mismo modo que un libro de texto difunde las enseñanzas de un profesor experto entre decenas de miles de alumnos, en lugar de hacerlo entre los pocos cientos a quienes podría enseñar personalmente en un año, las lecciones por computador serían aún de más valor. Podrían involucrar al alumno, examinarlo y permitir que se calificase a sí mismo según sus progresos y capacidades. El inconveniente hasta ahora es que los estudiantes no son, casi por definición, demasiado ricos, ya sea personalmente o en términos de los equipos que la sociedad está dispuesta a comprarles; por tanto, sólo pueden costearse computadores baratos que imponen todas las limitaciones que hemos señalado anteriormente. En consecuencia, el resultado es un software ordinario, que todavía no puede imitar el contenido y riqueza de un libro de texto común.

Un problema más serio es que enseñar consiste en algo más que obligar a los alumnos a leer los libros de texto y realizar pruebas para valorar el nivel de comprensión que han alcanzado. Un buen maestro tiene una relación mucho más íntima con la clase. Conoce lo que sus alumnos creen que saben y también lo que realmente saben. El profesor entiende los problemas que tiene cada alumno para aprender y, de acuerdo con esto, prepara la lección. Un buen profesor aprende de los alumnos a la misma velocidad con que éstos aprenden el tema. Para imitar esto es necesario un software de inteligencia artificial mucho más sofisticado, que sólo ahora empezamos a saber cómo escribir. Por tanto, no sorprende que los computadores jueguen un papel limitado en la enseñanza de materias escolares ordinarias.

Evidentemente, se empieza a pensar que la enseñanza de los computadores y de la programación es en sí misma una materia que los niños deben aprender y, precisamente, para enseñarla se necesitan computadores pequeños y baratos. De forma bastante curiosa, hay profesores que se resisten apasionadamente a su implantación. Sospecho que la razón está en que los niños no necesitan mucha instrucción formal para aprender el funcionamiento de un computador. La respuesta inmediata que obtienen de la máquina les entusiasma y les estimula a intentar más y más cosas, de manera que el profesor ve que su papel se reduce al de consejero ocasional para quien ya sabe lo que está haciendo. Algunos maestros encuentran este papel bastante humillante.

Casi todas las máquinas pequeñas se suministran con una versión de BASIC y los usuarios que se cansan de los juegos ya escritos, tratan naturalmente de escribir los suyos propios. Una tercera categoría de software para los computadores personales, directamente utilizable en las clases de “informática”, son versiones de los lenguajes de grandes máquinas tales como Lisp y Forth.

Por último, existen diversos intentos de proporcionar software de empresas tales como procesamiento de textos, cálculos de operaciones bursátiles, contabilidad y gestión de base de datos.

A pesar de las limitaciones de hardware, este mercado continúa creciendo. Tres años después de la aparición del primer microcomputador verdaderamente barato —el Sinclair ZX80—, sólo en el Reino Unido había varios millones de computadores personales, lo que según algunos periodistas entusiastas convertían dicho país en el más densamente informatizado del mundo. A mediados de 1983 se creía que uno de cada cinco hogares británicos tenía un microcomputador y el mercado de software empezaba a competir con el de la música pop o de las cintas de vídeo. De hecho, una de las principales compañías musicales, Virgin Records, se lanzó al negocio de los juegos de computador porque pensaron que era un área que no podían permitirse ignorar. Por otra parte, las fotografías de millonarios adolescentes que habían abandonado la escuela, pero que ahora estaban amasando grandes fortunas gracias a sus juegos de monstruos, se convirtieron en una imagen cotidiana de los periódicos. Como en cualquier mercado de consumo masivo, se insistió más en la presentación y propaganda que en la búsqueda de la perfección técnica.

Media un gran abismo entre la población de excéntricos aficionados a los computadores, que en 1979 luchaban con montones de tableros de circuitos y códigos máquina y la actual.

Resulta del todo evidente que no tardaremos mucho en ver cómo el microcomputador personal se convierte en una máquina de 16 bits con discos. Debería tener gráficos de alta resolución y gran volumen de RAM. La enorme cantidad de estas máquinas que se venderá hará que sus precios no sean mucho más altos que los actuales. Es de esperar que, con la introducción de grandes cantidades de lo que ahora consideramos como software profesional de alto precio, el mercado sufrirá una nueva convulsión y se distribuirá a precios muy por debajo de lo que cuesta en la actualidad.

JUEGOS DE COMPUTADOR

Para muchas personas los microcomputadores existen sólo para jugar. Este punto de vista quizá sea bastante limitado. Para otras, en cambio, la frase "juegos de computador" las sume en el desánimo; pero, tal vez, estas últimas también estén pecando de estrechez de miras.

Los computadores ofrecen un vehículo excelente para ciertos tipos de juegos debido a que resulta relativamente fácil construir a partir de ellos máquinas muy complicadas. Si se piensa en las tendencias sádicas de los autores de los juegos, quizá se considere una suerte que éstos no permitan más que un mínimo de participación física; sin embargo, la destreza y el esfuerzo de la mano y la vista deben ser tan grandes como en los más emocionantes juegos de pelota.

Hay un amplio espectro de estos juegos: desde las fáciles diversiones de los juegos de galería, como el de los "Invasores del espacio", a los elevados ejercicios intelectuales del ajedrez. Entre ambos extremos se encuentran algunas distracciones inherentes que no sólo exigen reflejos rápidos, sino también agudeza de pensamiento. A menudo estos juegos sitúan al jugador en la posición de un jefe militar que debe tomar decisiones estratégicas y tácticas luchando contra el tiempo, sin tener —tal y como tantas veces ocurre en la vida real— completo conocimiento de los hechos.

Si un día se escribiese la historia de los juegos de computador, se diría que tienen dos raíces distintas. La primera está en las indebidas diver-

siones de los programadores de los grandes sistemas comerciales y académicos, quienes, para mitigar la pesadez del trabajo rutinario, escribían juegos en las horas de comida. El venerable paquete Startreck es, posiblemente, el antepasado de todos ellos, y difícilmente habrá un solo gran computador en el mundo que no tenga este programa oculto en algún lugar de sus archivos. Los empresarios sensatos tienden a alentar el juego a un coste de miles de dólares por minuto, porque hace que los programadores se interesen en los instrumentos de su trabajo y les anima a experimentar.

El otro y más respetable antepasado de los juegos de computador es la simulación o modelo (véase p. 126). Los militares utilizan a menudo las técnicas del simulacro para predecir las consecuencias de nuevas armas y estrategias. En muchos de estos modelos la máquina se utiliza simplemente como un contable grande y rápido, que calcula los resultados de las distintas decisiones estratégicas posibles.

El peligro con las simulaciones reside en que la gente tiende a creérselas, incluso cuando se ha tomado un atajo en la lógica del sistema que le hace perder cualquier sentido.

Uno de los principales ejemplos de un caso de este tipo es la simulación de la economía mundial que encargó el Club de Roma (grupo de grandes hombres de negocios, políticos y funcionarios) en la década de los setenta. Se obtuvo un informe llamado Los *límites del crecimiento*, que predecía un colapso súbito de la economía mundial hacia el año 2000. Según éste, tanto los ricos como los pobres tendrían que enfrentarse con el hambre y el caos. Debido a que esta horrible predicción provenía de un grupo de personas de considerable prestigio, que aumentó todavía más gracias a la utilización de un computador, muchas personas se la creyeron a pies juntillas.

Más tarde se descubrió que en el modelo de computador había un fallo funesto. Tenía una importante sección sobre el precio y suministro de petróleo. Como todos sabemos, existe una cantidad limitada de petróleo en la Tierra, y no tardaremos mucho en vernos obligados a pensar en utilizar alguna otra fuente de energía. El programa reflejaba este hecho, pero no consideró en el modelo lo que ocurriría con el precio del crudo a medida que los pozos se fueran secando completamente. Suponía que el petróleo continuaría costando lo mismo que a inicios de los años setenta,

hasta que un día se acabaría. Naturalmente, esto hacía que la economía mundial se sumiese en la confusión.

En la vida real, a medida que las reservas de petróleo se terminen, las naciones productoras aumentarán los precios para compensar la disminución de sus ingresos en el futuro. El aumento del precio del petróleo estimulará automáticamente el desarrollo de fuentes de energía alternativas. Considerando tan sólo una eficacia moderada de los mecanismos de mercado, podríamos hacer que la transición del petróleo a cualquier otra fuente de energía que venga después, provoque únicamente una moderada dislocación y no el completo desastre profetizado por el modelo de computador.

Hay tres clases principales de juegos de computador: los nerviosos, los románticos y los intelectuales.

Los juegos nerviosos se denominan a menudo juegos de arcade; exigen rapidez en la vista y la mano y no dan mucho que pensar. Tienen una extraordinaria fascinación debido al *feedback* instantáneo: el lazo nervioso mano-ojo se extiende más allá del propio cuerpo para pasar a la máquina, manteniendo a los devotos de los juegos de arcade en una fuerte adicción.

Los juegos románticos derivan de un género de ficción basado en la obra de J. R. R. Tolkien *El señor de los anillos.* Se ampliaron a juegos de tablero y ahora han sido automatizados en una clase de juegos de computador llamada generalmente “aventura”. El núcleo de un juego de aventuras es algún tipo de paisaje invisible (que puede ser de tres dimensiones) dividido en celdas. Cada jugador asume un papel (mago, guerrero, duende, hechicero), reúne varias herramientas más o menos útiles (linterna, espada mágica, llave, frasco de agua) y empieza su investigación. En cada etapa puede ir arriba, abajo, hacia el este, oeste, norte o sur, entrando en otra celda. El camino de entrada a una celda puede ser una puerta estrecha que no permitirá pasar la bolsa llena de oro que el jugador “lleva” consigo, o que precisa de una llave para abrirla que el jugador no posee.

Cuando consigue entrar en la celda, ésta puede contener un grifo comedor de hombres o una hermosa joven. Los juegos de aventuras pueden durar varios días y los más modernos tienen dibujos sorprendentes.

Un tercer tipo de juegos es el intelectual, en el que se pone mucho más énfasis en la estrategia que en la creación de ambiente. Un ejemplo

típico es el de “Arnhem”, juego de guerra en el que el jugador debe dirigir batallones en lucha contra los nazis. Estos juegos tienden a poner énfasis en el comportamiento dinámico; se necesita tiempo y esfuerzo para lograr que una división de Panzers se mueva; pero, una vez se ha conseguido, se necesita algún tiempo para detenerla. Conceptos como el de servobucle cerrado (véase p. 189) son de gran utilidad para los autores de juegos intelectuales.

“Hammurabi” es un juego interesante en el que se proporciona al jugador un país feudal para que lo gobierne. Al inicio de cada año tiene cierto número de campesinos que provienen del año anterior, menos los que han muerto o escapado y más los que han nacido. Tiene cierta cantidad de grano almacenado, menos el que se han comido las ratas, cierto número de soldados, castillos, mercados y palacios, y debe tomar decisiones sobre el nivel de impuestos y la corrupción en la administración, sobre la cantidad de grano que debe distribuir a los campesinos o guardar para semilla. Para ello puede contratar soldados o construir palacios. Además, sus vecinos (dirígidos por el computador) pueden atacar. Según el número de soldados de que dispone, ganará y obtendrá más tierra o será derrotado y perderá territorios. Las decisiones que se toman prolongan sus efectos muchos años después. Todo el proceso es muy complicado y fascinante.

Estos juegos serían mucho más interesantes si pudieran presentar una imagen animada del mundo que representan. Pero, como veremos en la página 171, las representaciones que parecen vivas exigen gran capacidad de memoria, y las máquinas de 8 bits están lejos de tener la suficiente. En los juegos y simulaciones las ventajas de la capacidad de memoria de las máquinas de 16 bits resulta evidente. La simulación Microsoft de un avión ligero, en el computador personal IBM, es sorprendente: da una visión de los instrumentos del avión (todos funcionando como si fueran de verdad) y de la escena que se ve desde la cabina de pilotaje. Es tan realista que llega a ser tan aburrida como volar en un avión pequeño... a menos que se quede sin gasolina y se estrelle.

2. La programación

LA PROGRAMACIÓN

Hay una antigua leyenda sobre Alejandro Magno que los maestros de escuela acostumbraban a contar a sus alumnos y puede que aún lo hagan. Cuando era joven, apuesto, poderoso y monarca de todo el orbe conocido, Aristóteles le enseñaba matemáticas. Un día, cansado del esfuerzo que significaba estudiar geometría, preguntó si no había un método más fácil. Aristóteles le contestó que no había un camino real que condujese a la geometría; todos, ricos o pobres, fuertes o débiles, debían aprenderla de la misma manera.

Lo mismo ocurre con los computadores. Creo que, a medida que transcurra la década de los ochenta, se hará más y más necesario conocer el mundo de los computadores, así como después de la introducción de la imprenta fue esencial aprender a leer. Los analfabetos se encontraban en desventaja y, por ley natural, sucumbieron.

Informatizarse significa, en primer lugar, familiarizarse con la máquina, el teclado y las convenciones del tema: por ejemplo, pulsar “RETURN” para que ocurran ciertas cosas. En segundo lugar, significa entender todo un conjunto de conceptos nuevos para relacionarse con los elementos que se encuentran en el interior de la máquina y que no pueden verse, tales como ubicaciones de memoria de archivos, programas y datos. El problema es que únicamente puede descubrirse lo que se hace mal mediante los propios errores.

Los computadores sólo pueden hacer un número limitado de cosas, y gran parte de lo que hacen es interno. En lo que concierne al mundo exterior, la gran mayoría se limitará a escribir letras o hacer dibujos en una pantalla o sobre un papel. Para comunicarse con los computadores hay que usar un teclado o una palanca de mando.

En términos humanos, un computador es un inválido que puede escribir y dibujar pero no moverse, coger algo o, incluso, ver lo que pasa a su alrededor. Así, la primera tarea en la programación es ser realista sobre lo que puede hacer. La gente piensa a menudo que puede utilizar un computador en su hogar para hacer cosas tales como los menús. Es

una buena idea, pero poco realista. Les gustaría que mirase en la despensa y en el frigorífico e informase sobre las existencias de la casa en un momento dado, como tres latas de sardinas, la mitad de un tarro de mahonesa y algo de arroz, de manera que la mejor comida que puede prepararse es una especie de ensalada. «¿Qué pasa con la lechuga?», le preguntarían, y el computador contestaría diligentemente que el gato se ha ensuciado en ella.

Sin embargo, esto es lo que el computador no puede hacer. El pobre aparato, sin piernas ni oídos, no puede fisgonear en la despensa. El único modo de introducirle información consiste en que usted se convierta en sus ojos y sus piernas, y teclearle después los datos. Mientras hace todo esto, su propio computador, que es un perfecto productor de menús, guiado por el hambre —a diferencia de lo que ocurre con el computador de silicio—, habría planeado una comida y ya la tendría a medio hacer.

La esfera práctica de acción de un computador corriente se limita al manejo de símbolos en una pantalla y sobre papel, lo que resulta suficiente, ya que cubre gran parte de la vida intelectual, en los negocios, ciencias y diversiones.

El segundo paso en la escritura de un programa consiste en encontrar algunos datos a los que tanto la máquina como usted pueden darles algún significado.

El juego de los “Invasores del espacio” es un excelente ejemplo: la gente ve las distintas formas que aparecen en la pantalla como extraños monstruos que deben ser eliminados; el computador los ve como pequeñas formas bien hechas que pueden dibujarse aquí o allí a gusto del jugador. Cuando las coordenadas del misil coinciden con las coordenadas del monstruo, la máquina reemplaza el monstruo por la figura de una explosión y envía un tono musical al altavoz.

En los negocios, la gente alimenta la máquina con listas de números que para ellos significan entradas o salidas de dinero (pérdidas o ganancias), éxito o desastre; para el computador son simples números que saben perfectamente cómo manejar. Si las deudas son mayores que los ingresos, la ganancia tiene el símbolo esto es todo y a la máquina le importa muy poco si el propietario se desespera.

De este modo, siempre que se hayan escogido algunos símbolos convenientes para ambos, nos debemos preguntar si las instrucciones del computador pueden hacer algo útil con ellos. Los lenguajes que se ejecu-

tan en los microcomputadores son muy inteligentes en ciertos sentidos y muy estúpidos en otros. Operan con logaritmos y senos y cosenos, lo que muchos adultos inteligentes no son capaces de hacer; pero, en cambio, no encuentran cómo organizar un problema.

Aunque hay muchas cosas que un computador no puede hacer, hay otras muchas que sí puede, siempre que nos tomemos las molestias suficientes. Veamos lo que se debería hacer si queremos que un computador encuentre el número de días que hay entre dos fechas determinadas cualesquiera. Parece fácil, pero de hecho requiere un programa bastante inteligente.

Para empezar debemos recordar que los americanos escriben las fechas en forma inglesa antigua, mes, día, año; mientras que los europeos modernos, lo hacen en día, mes, año. Una vez se le dice al programa cuál de las dos escrituras se prefiere, ha de tener la versatilidad necesaria para manejarse con ambas. (Así, se le dice que “4.3.84” significa 3 de abril para un americano y 4 de marzo para un europeo y no existe ninguna forma de que la máquina pueda deducir cuál es cuál.)

Debe ser capaz de manejar fechas tales como “3 de abril de 1982” o “16/5/79”. No debería importarle si se utilizan espacios, puntos, rayas o barras entre los números. Si se ha dado un nombre al mes (recordemos que el programa maneja un número mínimo de letras, tanto mayúsculas como minúsculas, necesarias para una identificación unívoca: “Ag” por agosto, “Jul” por julio), ha de transformar el nombre en un número. Luego, debe saber cuántos días han pasado desde el inicio del año hasta el primero de este mes. Durante el proceso debería detectar fechas imposibles, como “31/4/1983”, o “30 Feb 1600” y adivinar que “83” significa “1983” y no un momento de la historia de Roma.

Asimismo tiene que calcular el número de días que hay entre cada una de las fechas y una fecha determinada del pasado lejano. Para prevenir eventuales sorpresas en algunos casos se hace coincidir esta fecha con la del inicio del calendario actual en 1582. Este proceso tiene en cuenta la variación del número de días en los meses, un día extra en un año bisiesto (divisible por 4) y el día extra anulado en un año superbisiesto (divisible por 400). Luego, cada fecha será transformada en un número de día.

Concluidos los cálculos que nos dan estos dos números de día para las dos fechas, resulta fácil comprobar cuál de los dos está en primer lugar mirando simplemente cuál de los números de día es menor, y para

que sepamos cuántos días hay entre las dos fechas sólo se precisa restar uno del otro. Todo este asunto supone recurrir a la biblioteca para comprobar los años bisiestos, así como cerca de dos días de dura programación. Sin embargo, en ocasiones, tanto usted como yo, podemos hacerlo de cabeza. Los computadores son muy pero que muy estúpidos.

Escribir un programa como éste se encuentra fuera del ámbito de este libro. Por tanto, vamos a considerar una tarea mucho más sencilla utilizando un lenguaje hasta ahora desconocido que me he inventado, llamado "FOODGOL". Con él me propongo ilustrar los conceptos básicos de la programación: pasos *(steps),* pruebas *(tests)* y bucles *(loops).*

Pasos, bucles y subrutinas

Como ejemplo sencillo, veamos los pasos que se necesitan para comer el almuerzo:

Programa para comer 1

1. Coja cuchillo y tenedor
2. Corte un pedazo de comida
3. Pínchelo con el tenedor
4. Introdúzcaselo en la boca
5. Corte un pedazo de comida
6. Pínchelo con el tenedor
7. Introdúzcaselo en la boca
8. Corte un pedazo de comida
9. Pínchelo con el tenedor
10. Introdúzcaselo en la boca
11. Corte un pedazo de comida
12. Pínchelo con el tenedor

Parece bastante razonable, aunque algo aburrido. Pero podemos simplificar mucho el programa incorporando un bucle:

Programa para comer 2

1. Coja cuchillo y tenedor
2. Corte un pedazo de comida
3. Pínchelo con el tenedor

4. Introdúzcaselo en la boca
5. Goto (acuda a) 2

Ahora, cada vez que introduzca la comida en la boca, el programa le hará volver al paso 2: «Corte un pedazo de comida», seguido de «Pínchelo con el tenedor», etc. Espere. ¿Qué ocurrirá cuando se haya comido toda la comida del plato?

Programa para comer 3

1. Coja cuchillo y tenedor
1a ¿Hay comida en el plato? En caso negativo deje el cuchillo y el tenedor, pare
2. Corte un pedazo de comida
3. Pínchelo con el tenedor
4. Introdúzcaselo en la boca
5. Goto 1a

Lo que faltaba era un test para ver si la acción debía continuar. Ahora tenemos la programación completa de un bucle. Procederá de forma que usted empiece a comer, continúe comiendo y pare de comer cuando se haya terminado el plato. De hecho, resulta difícil ver cómo podría escribirse el programa sin utilizar un bucle. Mire de nuevo el programa para comer 1. ¿Cómo sabe por adelantado cuántas veces ha de repetir el ciclo? ¿Escribe 10 ciclos pero no confía que le sirvan un banquete? ¿O dicta 100, y quizá nos encontremos arañando un plato vacío docenas de veces? Ha de existir un test que compruebe si el ciclo ha terminado.

Podemos alargar el programa para que pueda comer algo de postre.

Programa para comer 4

1. Coja cuchillo y tenedor
1a ¿Hay comida en el plato? En caso negativo, deje el cuchillo y el tenedor, goto 6
2. Corte un pedazo de comida
3. Pínchelo con el tenedor
4. Introdúzcaselo en la boca
5. Goto 1a
6. Pida el postre

7. ¿Hay comida en el plato? En caso negativo deje el cuchillo y el tenedor, pare
8. Corte un pedazo de comida
9. Pínchelo con el tenedor
10. Introdúzcaselo en la boca
11. Goto 7

Hemos modificado el programa de tres maneras. Primera, cuando se ha terminado con el primer plato, el paso la nos lleva al paso 6, que nos hace pedir el postre. Hay un nuevo bucle para comer el postre, con un nuevo test para ver si se ha terminado.

Sin embargo, las líneas, 8, 9 y 10 son copias exactas de las 2, 3 y 4. Las líneas 11 y 5 son muy parecidas. Podemos hacer la misma parte del programa de test, comiendo los dos platos mediante una “subrutina”.

Programa para comer 5

1. Coja cuchillo y tenedor
1a ¿Hay comida en el plato? En caso negativo deje el cuchillo y el tenedor, goto 6
2a Gosub 100
5. Goto la
6. Pida el postre
7. ¿Hay comida en el plato? En caso negativo, deje el cuchillo y el tenedor, pare
8. Gosub 100
11. Goto 7
100. Corte un pedazo de comida
101. Pínchelo con el tenedor
102. Introdúzcaselo en la boca
103. Return (retorno)

“Gosub” es una instrucción que hemos tomado prestada de BASIC y que significa “Ir al paso ordenado (en este caso 100), hacer los pasos que siguen, y volver cuando se encuentre la orden RETURN”. Después de haber saltado desde el paso 2a a la subrutina en el paso 100, ejecutando allí los pasos y vuelto, el programa hace el próximo paso, que es el 5, igual que antes. Cuando el control en la indica que se ha terminado el

primer plato, el programa salta —igual que antes— al paso 6, y utiliza la subrutina en el paso 100 para comer el postre.

¿Para qué sirve todo esto? En primer lugar, ahorra espacio del programa. En segundo, significa que el mismo trozo de programa realiza toda la acción de comer. Si esta acción tuviera que hacerse de otro modo, utilizando, por ejemplo, palillos en lugar de cuchillo y tenedor, puede cambiarse fácilmente el modo de funcionamiento del programa. En este caso debemos hacerlo, porque nos hemos olvidado de indicar al desafortunado comensal que mastique y trague la comida. Mientras el programa esté en acción, irá llenando su boca de comida hasta ahogarse.

Programa para comer 6

1. Coja cuchillo y tenedor
la ¿Hay comida en el plato? En caso negativo, deje el cuchillo y el tenedor, goto 6
2a Gosub 100
5. Goto la
6. Pida el postre
7. ¿Hay comida en el plato? En caso negativo, deje el cuchillo y el tenedor, pare
8. Gosub 100
11. Goto 7
100. Corte un pedazo de comida
101. Pínchelo con el tenedor
102. Introdúzcalo en la boca
102a Mastique
102b Trague
103. Return

Habría sido mucho más difícil alterar el Programa para comer 1; tendríamos que haber añadido dos pasos nuevos cada tres pasos y fácilmente nos habríamos equivocado. También hemos previsto un “bug” (error), trozo de programa que no hace lo que le correspondería. Los programadores profesionales pasan la mayor parte del tiempo haciendo esto.

BASIC

En el apartado anterior vimos el modo como se escriben los programas en muchos de los lenguajes existentes; se da una lista de órdenes que deben ser ejecutadas una tras otra; se condensan acciones similares en rutinas únicas que son solicitadas en distintos lugares del programa y se controla el resultado de esta manera obtenido para ver si se han realizado las diferentes etapas.

Esta clase de estructura de programa es utilizada por una amplia gama de lenguajes: Fortran, Algol, COBOL, Pascal, “C”, entre muchos otros, y también BASIC. El BASIC será, probablemente, el primer lenguaje con el que se encontrarán la mayoría de lectores de este libro, debido a que una amplia mayoría de computadores personales tienen instalados dialectos suyos. Tiene desventajas, con las que ya nos encontraremos, pero también tiene grandes ventajas.

La más importante es que resulta sencillo empezar con BASIC. Casi todo el mundo puede escribir algún tipo de programa al cabo de unos pocos minutos de estar frente a un computador, aunque sólo sea escribir “Hola” en la pantalla. Por otra parte, el BASIC es interpretado. Esto significa que puede escribirse un trozo de programa y pasarlo inmediatamente para ver si funciona. Otros programas más serios deben compilarse cada vez antes de que se puedan pasar. Este proceso puede durar desde algunos minutos hasta horas y resulta muy descorazonador para el principiante. Si su programa en BASIC no funciona (como le ocurrirá la mayoría de las veces), se edita y se intenta de nuevo.

Entre los inconvenientes destaca el que no exista un BASIC estándar. En el Apéndice 1 se da una lista de las instrucciones en una de sus versiones más populares; le sugiero que la utilice como ayuda para seguir lo que haremos a continuación.

El lenguaje BASIC consiste en tres tipos de elementos: instrucciones, variables y funciones. Las instrucciones son fáciles de entender. Hay verbos: ‘PRINT’, imprimir algo en la pantalla; ‘LPRINT’ imprimir algo en la impresora; V, para sumar dos números entre sí, y así sucesivamente. Las variables se parecen mucho a las que encontramos en el álgebra, excepto que en lugar de ‘x’ e ‘y’, en muchos dialectos de BASIC pueden dárseles nombres más útiles como HORASPORSEMANA o KLINGONSMUERTOS. Existen dos tipos de variable, una para representar un

número y otra para representar un texto. Si el programa tuviera que preguntar el nombre e ingresos del usuario escribiríamos las líneas:

```
10 PRINT "Por favor escriba su nombre"
20 INPUT NOMBRE$
30 PRINT "y sus ingresos"
40 INPUT INGRESOS
```

Este programa es, de hecho, muy pequeño, pero hay varias cosas a tener en cuenta. Primero, las líneas están numeradas y el programa ejecutará las instrucciones que se indican siguiendo el orden de los números. Segundo, los números van (convencionalmente) de 10 en 10, de manera que se pueden insertar más tarde líneas extras si así se desea. Tercero, a la orden PRINT le sigue un *string*, es decir, algunos caracteres entre comillas, o una variable (véase más abajo). Y, cuarto, la orden INPUT espera que el usuario escriba algo en el teclado, seguido de un RETURN. Cuando se pase este programa, veremos en la pantalla:

«Por favor escriba su nombre»
?

(INPUT coloca un ? para indicar que espera que se haga algo.)

Se escribe Luis o el nombre que sea y se pulsa RETURN. El BASIC sabe ahora que el "valor" de la variable NOMBRES es "Luis", La '$' después de la variable NOMBRE sirve para indicar al BASIC que lo que sigue debe ser tratado como texto y no como número; en el interior del computador cada uno de ellos se trata de modo muy distinto.

Entonces el programa escribirá en la pantalla:

«y sus ingresos»
?

y se escribe 14 o 14.000 o la cantidad que sea, seguido de RETURN; este número será ahora el valor de la variable INGRESOS. Puesto que el BASIC empezó como lenguaje para hacer matemáticas, ya supone que una variable sin ninguna que la distinga es un número.

Debido a que los programas hacen muchas cosas de este tipo, el lenguaje proporciona un método más compacto; se puede construir un *prompt* dentro de la instrucción INPUT:

10 INPUT “Por favor entre su nombre e ingresos separados por una coma”; NOMBRES, INGRESOS

La serie de caracteres entre comillas colocada después de INPUT queda impresa en la pantalla. Le sigue un ‘;’ para indicar al BASIC que no mueva el cursor a una nueva línea, lo que haría en caso contrario después de imprimir cualquier cosa. Entonces espera que el usuario escriba:

Luis, 14.000

No olvidemos que INPUT espera una coma que no podemos omitir, ya que en caso contrario BASIC se quejaría produciendo un “Error message” (mensaje de error). Si intentamos entrar: «Luis 14.000» o «Luis -4.000», el BASIC no lo entenderá.

Hasta ahora hemos alimentado nuestro programa con dos variables: una alfanumérica y otra numérica. Una vez introducidas, podemos hacer varias cosas y, siguiendo con estos dos tipos de datos, podemos llevar a cabo dos clases de operaciones.

Primera, echemos una mirada a las operaciones con conjunto de caracteres. Sería agradable que el computador respondiera elegantemente saludando a la persona que acaba de escribir un nombre. Podríamos dictarle:

«Buenos días Luis. Encantado de conocerle.»

Hay varias formas de hacer este truco. Una es imprimirlo todo en la pantalla con un programa como el siguiente:

```
20 ? «Buenos días»;
30 ? NOMBRE$;
40 ? «.Encantado de conocerle.»
```

El ‘?’ es un símbolo taquigráfico por PRINT en muchos BASIC. Originalmente se utilizó para imprimir variables desconocidas, de manera que tuviera algún sentido desde el punto de vista del usuario: «Qué es (?) la variable fulano de tal», tal como se utiliza aquí en la línea 30. El punto después de “Luis” debe ir en la tercera línea. Recuerde que el ‘;’ impide

la aparición de nuevas líneas después de cada línea de programa. Sin él veríamos:

Buenos días Luis
.Encantado de conocerle

Otra manera de hacerlo sería “concatenar” estas tres variables alfanuméricas de modo que formasen una variable alfanumérica única y a continuación imprimir esto:

```
20 B$=«Buenos días»+NOMBRE$+«.Encantado de conocerle.»
30 ?B$
```

Hemos construido una nueva variable, B$, a partir de los tres trozos que teníamos. El signo ‘+’, cuando se utiliza con variables alfanuméricas significa “pegar entre sí”. También se dará cuenta que el ‘=’ se utiliza de una forma extraña (evidentemente desde el punto de vista del álgebra). Es una pena que el BASIC utilice el ‘=’ en dos sentidos bastante distintos, lo que puede confundir fácilmente a los principiantes.

Tal como lo hemos utilizado aquí, significa “transfiera lo de la derecha a la izquierda”, o “iguale la parte izquierda con la derecha”. Quizás habría sido mejor utilizar un símbolo como ‘←’ para sugerir el movimiento hacia la izquierda; aunque habríamos tenido que pulsar un carácter extra cada vez.

Algunos lenguajes utilizan el símbolo ‘:=’ para distinguirlo del verdadero igual:

SI A=B ENTONCES... (IF A=B THEN...)

Podríamos utilizarlo aquí. Podría haber una persona en particular que no nos gustase, de manera que podríamos introducir una nueva línea (¿se da cuenta de la utilidad de numerar las líneas de 10 en 10?):

```
15 IF NOMBRE$=«Juan» THEN «Lárguese»:GOTO 10
```

Este es un igual verdadero y comprobará si el odiado Juan se encuentra frente al teclado. (En la práctica deberíamos comprobarlo tanto para “juan” como para “JUAN”.) La instrucción ‘IF’ corresponde al si condicional castellano y funciona en el mismo sentido. Si el NOMBRE$ es Juan, entonces el programa ejecuta la instrucción después de ‘THEN’; en

caso contrario ignora esta línea y “se viene abajo” hasta la línea 20 como antes.

Nos encontramos con un nuevo símbolo, ‘:’. Este símbolo introduce una nueva orden después de la orden de imprimir “Lárguese”, para regresar de nuevo a la línea 10, que solicita a otra persona que introduzca su nombre e ingresos.

Matrices

No se puede llegar muy lejos en BASIC (o en cualquier lenguaje informático) sin chocar con la idea de matriz. Una matriz es una variable con una serie de “casillas” que permiten guardar varias cadenas o números en lugar de uno solo.

El tipo de matriz más simple, llamado “vector”, tiene sólo una fila de posiciones de la variable. Supongamos que estamos interesados en el tiempo y queremos registrar la lluvia total caída cada día de la semana. Lo que necesitamos es justamente un vector. Al principio del programa debemos componer la matriz con la orden de dimensión, ya que el BASIC debe reservarle algún espacio:

```
DIM LLUVIA(7)
```

Esto le dice al programa que queremos una variable denominada “LLUVIA” con siete casillas a las que podemos acceder llamando (de manera bastante natural) a LLUVIA(l), LLUVIA(2), etc. De este modo podemos escribir un pequeño programa que ejecutaremos en la máquina el domingo, y que nos permitirá introducir la cantidad de lluvia que hemos registrado durante la semana:

```
10 DIM LLUVIA(7)
20 FOR K=1 TO 7
30 INPUT «Lluvia caída hoy»;LLUVIA(K)
40 NEXT K
```

Este programa utiliza un bucle constituido por las líneas 20 y 40. El bucle empieza con la variable K colocada en 1, y aumenta de uno en uno hasta que llega a 7. Cuando se pasa el programa, nos pregunta «¿Lluvia

caída hoy?» y entonces introducimos 2,3 o 0,006 o el valor que sea, y este valor queda colocado en la casilla apropiada de LLUVIA.

En la siguiente ilustración vemos que el lunes fue húmedo, mientras que el sábado y domingo hubo lluvia torrencial.

		1	2	3	4	5	6	7
Lluvia		0,9	0,01	0,04	0,6	1,2	4	2
							▼	
Día		Lun	Mar	Miér	Jue	Viern	Sáb	Dom
lluvia	1	9	01	0,04	0,6	1,2	4	2
T máx	2	14	22	24	20	17	16	18
T mín ►	3	0,6	9	10	8	7	7	7
humedad	4	0,7	0,6	0,5	0,55	0,6	0,8	0,7
lluvia	5	1	8	9	5	1	0	0
viento	6	10	2	2	15	20	25	20

En BASIC también pueden tenerse matrices de caracteres o alfanuméricas. Sería magnífico que este pequeño programa nos pidiese la lluvia caída diciéndonos el día de la semana. Para conseguirlo necesitamos establecer la correspondiente matriz de variables alfanuméricas llamadas DIAS, en la que almacenamos “Lunes”, “Martes”, “Miércoles”, “Jueves”, etc.

```
10 DIM LLUVIA(7),DIAS$(7)
20 DATA Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo
30 FOR K=1 TO 7
40 READ DIA$(K)
50 NEXT K
60 FOR K=1 TO 7
70 ? «Lluvia caída»;DIA$(K):INPUT LLUVIA(K)
80 NEXT K
```

La línea 20 retiene los días de la semana como datos, separados por comas. La orden READ (leer) en la línea 40 los introduce de uno en uno en DIA$(K), con lo que ahora tenemos una segunda matriz.

El segundo bucle, en la línea 60, imprime el nombre apropiado del día en la línea 70, y solicita que se escriba la lluvia caída, igual que antes.

Podríamos haber combinado los dos bucles, pero resulta más fácil ver lo que pasa de este modo.

Hay un tipo de matriz más completo que retiene varias filas de variables, tanto alfanuméricas como numéricas. Si tuviéramos una estación meteorológica y quisiéramos almacenar media docena de números distintos para cada día: lluvia caída, horas de insolación, temperaturas máxima y mínima, humedad y velocidad del viento, deberíamos dimensionarlo al inicio del programa con:

```
1 DIM MET(7,6)
```

y el modo de acceso sería el mismo. Por ejemplo, la temperatura mínima del viernes se almacena en MET(7,3) y tiene como valor 7.

Números

En las páginas anteriores hemos visto algunas de las cosas que el BASIC puede hacer con variables alfanuméricas. Ahora echaremos una mirada a los números. Antes solicitamos que se introdujesen el nombre y los ingresos de la persona que estaba frente al teclado, y ambas cosas se introdujeron como variables. Tenemos los ingresos en la variable numérica INGRESOS, y podemos realizar distintos cálculos con ella.

```
10 INPUT «Por favor entre su nombre e ingresos, separados por una
   coma»; NOMBRE$,INGRESOS
15 IF NOMBRE$=«Juan» THEN ?«Lárguese»;GOTO 10
20 B$=«Buenos días»+NOMBRE$+«.Encantado de conocerle.»
30 ?B$
40 MIOS=INGRESOS•.8
50INGRESOS=INGRESOS–MIOS
60 ?MIOS;«de éstos serían míos;»;INGRESOS;«de éstos serían tuyos»
```

Hemos añadido tres líneas de programa nuevas: 30, 40 y 50, que podrían parecer bastante enigmáticas. La línea 40 calcula una nueva variable, MIOS, como el 80% (0,8) de la variable INGRESOS. El símbolo '•' se utiliza para multiplicar, ya que 'x' podría confundirse con una variable.

La línea 50 tiene sentido si recordamos que significa “póngalo de la derecha a la izquierda”. Esta línea dice “haga que el nuevo valor de los INGRESOS sea el valor antiguo, menos MIOS”. La línea 60 de la salida impresa da la codiciosa visión que el pequeño computador tiene de la situación económica:

«11.200 de éstos serían míos; 2.800 de éstos serían tuyos»

Hay muchas otras cosas que el BASIC nos permite hacer con los números. Se puede restar y dividir, elevar a potencias, sacar logaritmos y antilogaritmos, calcular senos y cosenos y, a partir de ellos, todas las demás funciones trigonométricas. Se pueden generar números aleatorios, es decir, números producidos con un procedimiento lo bastante complicado para que parezcan aleatorios a un observador casual. Sin duda, un computador, al ser una máquina completamente lógica, no puede producir un verdadero número aleatorio, es decir, uno seleccionado completamente al azar. Lo mejor que hace es empezar la serie de números que parecen aleatorios con una “semilla” que introduce el operador. Cada vez que se ejecuta el generador del número aleatorio con el mismo dato “semilla”, producirá las mismas series. Cuando se necesita un número aleatorio verdadero (como en el sorteo de la lotería Nacional), es necesario que compute algunos acontecimientos subatómicos impredecibles como la desintegración de átomos radiactivos. Esto tiene interesantes implicaciones filosóficas sobre las diferencias entre los seres humanos (como fuente de aleatoriedad) y las máquinas. La mayoría de estas operaciones son realizadas por “funciones”, el tercer paquete de operaciones que nos ofrece ese laberinto que es el BASIC.

Para calcular el logaritmo de un número, A, simplemente se escribe:

```
70 B=LOG(A)
```

utilizando la función LOG. También hay funciones que se utilizan con variables alfanuméricas:

```
70 B=INSTR(B$,«1»)
```

nos encontrará a qué distancia dentro de B$ (tal como se ha definido en la línea 20 de nuestro pequeño programa) ha llegado la letra «I», y resultará ser 20.

Gráficos

Cada vez con mayor frecuencia los microcomputadores aparecen en el mercado provistos de pantallas de resolución bastante elevada y dotados de algunos gráficos primitivos (según los estándares CAD). El objetivo consiste en proporcionar como mínimo instrucciones para dibujar un punto, una línea recta o colorear un área. Un número determina el color de lo que se está dibujando. Por ejemplo, para dibujar un punto en la posición X,Y (las coordenadas que tienen su origen, 0,0, en la esquina inferior izquierda), diríamos:

```
PLOT POINT (C,X,Y)
```

Para dibujar una línea de color 5 desde el punto 34,56 hasta el punto 121,444, diríamos:

```
PLOT LINE (5,34,56,121,444)
```

La orden:

```
PLOT FILL (3,34,56,121,444)
```

llenaría el rectángulo definido por los dos puntos situados en los extremos de una de sus diagonales con el color 3.

En algunos BASIC con capacidad para trazar gráficos, no se tienen lujos tales como PLOT, LINE o FILL. Para dibujar una línea controlada por el programa, es necesario que se encuentre el lugar donde deben colocarse cada uno de los puntos. Por ejemplo, para dibujar una línea de cinco unidades de longitud, con una pendiente hacia abajo de 45 grados desde el punto X,Y, escribiríamos:

```
100 FOR K=l TO 5
110 FOR J=1 TO 5
120 PRINT AT X+K, Y–J, ‘•’
130 NEXT J
140 NEXT K
```

para producir este efecto:

```
•
 •
  •
   •
    •
```

Cambiando los signos en la línea 120, la línea puede ser trazada bajo cualquiera de las cuatro orientaciones. Excluyendo la coordenada X o la Y, se obtiene una línea vertical u horizontal. En otros ángulos las líneas son más difíciles de construir y se presentan de la siguiente manera:

```
                  ••••••
            ••••••
      ••••••
••••••
```

El programa tiene que calcular cuántos pasos horizontales se necesitan por cada paso vertical, o viceversa.

Hay muchas máquinas, particularmente las más serias construidas para usos empresariales, que no tienen instrucciones gráficas. Para imprimir un carácter en la pantalla en la posición X,Y, tienen que hacerlo del modo menos flexible. Cada máquina tiene un mando para enviar el cursor a "Home" (en general, el extremo superior izquierdo de la pantalla), de modo que allí es donde va a parar. Luego se mueve Y líneas hacia abajo y X espacios a la derecha y se imprime el carácter apropiado. Después se repite el proceso para el próximo carácter.

Nada de esto parece un método muy apropiado para dibujar una "Estación de combate Hyperon" con vistas a su próximo juego. El método más sencillo es utilizar un *digitizing pad* (véase p. 170), si se dispone de uno. En caso contrario, existe un modo de dibujar en BASIC utilizando instrucciones DATA. El siguiente programa dibujará un barco en cualquier posición de la pantalla:

```
10 HOME$=CHR$(29) 'REM EL CARACTER CORRESPONDIENTE
   A HOME
20 ABAJO$=CHR$( 10) 'REM CURSOR ABAJO
30 RT$=CHR$(32) 'REM CURSOR HACIA LA DERECHA
40 W=79:H=23 'REM LONGITUD Y ALTURA DE SU PANTALLA
```

```
200 INPUT "POSICION"; X,Y
210 INPUT "QUE SE IMPRIMA EL BARCO";I
220 ON I GOSUB 5000, 6000, 7000
230 A$=INPUT$(1)
240 'REM * * * * ** * * * * * ** * * ** * *
900 READ HT
910 IF Y+HT>H THEN HT=H–Y
920 FOR I=1 TO HT
930 READ D$
1000 PRINT HOMES
1010 PRINT STRING$ (Y+I, ABAJO$);
1020 PRINT STRING$ (X,RTS);
1030 PRINT LEFTS (D$, W–X);
1035 NEXT I
1040 RETURN
4998 'REM
4999 'REM * * * * ** * * * * * ** * * ** * *
5000 RESTORE 10000:GOSUB 900: RETURN
6000 'REM SALTAR A OTRO DIBUJO
10000 DATA 7
10010 DATA"    *      *
10020 DATA"    *      *
10030 DATA"    *   *****
10040 DATA"    *   *****
10050 DATA" ************
10060 DATA"  ***********
10070 DATA"    **********
```

La figura se ha dibujado tal como aparecerá en la pantalla en las instrucciones DATA de las líneas 10010-10070. La línea 10000 tiene una instrucción DATA numérica que dice al programa con cuántas líneas de dibujo debe contar. Esto permite añadir otros dibujos con más o menos líneas. Las líneas 10-40 preparan la pantalla para su máquina.

La línea 200 pide la posición en la pantalla, y las líneas siguientes tienen en cuenta que se podría tener que dibujar más de una figura (para lo cual aquí no hay espacio). La instrucción ON ... GOSUB envía la ejecución a la primera, segunda o tercera subrutina especificada.

La subrutina 5000 pone el mecanismo DATA en condiciones para leer en la línea 10000. Si se tiene otro dibujo en la línea 11000, entonces la subrutina 6000 sería RESTORE 11000 ... Entonces la ejecución salta a la línea 900. Las líneas 900 y 910 calculan cuánta parte del dibujo debe suprimirse si éste sobrepasa el borde de la pantalla. El bucle de las líneas 920 y 1040 imprime cada línea de datos del dibujo en el lugar apropiado de la pantalla, truncándolo si la rebasa.

Con la ayuda de estas escasas instrucciones BASIC, las personas ingeniosas pueden escribir programas para la visualización en la pantalla de la esfera de un reloj con las agujas en movimiento.

Si miramos el Apéndice 1, veremos que hay muchas instrucciones y funciones que no hemos tocado. Sin embargo, si realmente se quiere aprender BASIC, la única manera de hacerlo es comprarse un computador pequeño y trabajar en la práctica. Quien pretenda aprender cómo se programa sin un computador obraría de modo parecido a aquel que aprendiera a montar en bicicleta sin una bicicleta. Durante el proceso de aprendizaje de BASIC se tendrá también ocasión de aprender lo que denomino *computish:* la forma increíblemente estúpida como "piensan" los computadores. Y mientras se aprende esto, lo más probable es que se llegue a una valoración sensata de lo que pueden y de lo que no pueden hacer estas máquinas.

PROGRAMACIÓN ESTRUCTURADA

Quien se introduce en el mundo de la informática no tarda en oír o leer la frase "programación estructurada". Si se debería o no enseñar a los niños (y también a los adultos) a programar de este modo, es una cuestión harto polémica.

La esencia del problema está en el BASIC. El BASIC resulta fácil de aprender; pero, según dicen los defensores de los lenguajes de programación "reales", esto es tan sólo una ilusión. Es fácil de aprender porque en realidad no es un verdadero lenguaje de programación; es algo parecido a un estanque en el que se puede penetrar sin peligro y con comodidad pero que tiene poca profundidad, de modo que resulta imposible nadar en él.

Los lenguajes propiamente dichos, a los que se conoce por el nombre de "lenguajes estructurados", son más difíciles de aprender porque no

corresponden a nuestra experiencia ordinaria. Su esencia radica en que se puede seguir construyendo. Con ellos escribimos los programas en pequeños bloques lógicos de instrucciones denominados “procedimientos”. En un primer nivel, los procedimientos son como las subrutinas “para comer” que encontramos en nuestro programa para comer de las páginas 82-85. El BASIC hace que trabajemos como el constructor de una cabaña, que hace que le envíen todos los materiales al solar y, una vez reunidos, empieza a ensamblarlos. No tiene necesidad de un plan, simplemente se pone a trabajar de una manera sencilla y natural.

Sin embargo, no puede edificarse una catedral del mismo modo como se construye una cabaña. Hay que pensar como arquitectos en vez de como chapuceros. Se empieza con piedras imaginarias, que combinamos para construir arcos y pilares, para luego olvidarnos de ellas; a continuación se combinan los pilares y arcos para formar las bóvedas y nos olvidamos de ellos; luego combinamos las bóvedas, formando naves, y las naves entre sí y construimos una catedral. Probablemente, un arquitecto sería incapaz de hacer todo esto si se viese forzado a pensar en las piedras concretas en cada una de las etapas, cosa que hace el BASIC.

Existe un montón de lenguajes estructurados, que se dividen en dos grupos. Los más comunes son Fortran, Pascal, “C” y Algol, que a menudo se denominan lenguajes Von Neumann, en honor a este distinguido matemático cuyo nombre se asocia con los primeros desarrollos de la informática. Después están los lenguajes “procesadores de listas” como Lisp, y los lenguajes de enseñanza Logo y Prolog.

El Logo es interesante principalmente debido al apasionado fervor que muestran sus entusiastas. Se presenta normalmente en un microcomputador que dirige un mecanismo sencillo, pero fabricado con gran precisión, llamado tortuga, que realiza dibujos con un lápiz. La razón por la cual se ha hecho dibujar a la máquina es para que resulte apropiado para la enseñanza. La tortuga se desliza lentamente sobre una hoja de papel colocada en el suelo. Tiene un lápiz en su centro que puede ser empujado hacia abajo o hacia arriba. Además, puede ir hacia delante o girar sobre su centro. La tortuga está conectada por un cable al computador y responde a instrucciones tales como:

FORWARD:X (avanzar la distancia X)
RIGHT:A (girar hacia la derecha A grados)

DRAW: (hacer descender el lápiz)

Se escribe un programa definiendo palabras. Para dibujar un cuadrado podríamos empezar definiendo la palabra HOOK que consiste en mover la tortuga X unidades hacia delante (FORWARD) y luego hacerla girar hacia la derecha (RIGHT). Si hacemos esto cuatro veces obtendremos un cuadrado.

En poco tiempo, un programador Logo decidido habrá construido una gran cantidad de verbos que podrá combinar entre sí (lo mismo se aplica en Lisp y Forth).

LENGUAJES TRADICIONALES

Un lenguaje es simplemente una forma de comunicarse con un computador utilizando instrucciones precisas. El lenguaje es el resultado de un compromiso entre lo que la máquina reconoce y lo que las personas entienden.

El primer lenguaje, el código en lenguaje máquina (véanse pp. 104-120), era muy difícil. Los esfuerzos para resolver las operaciones aritméticas decimales en el sistema binario fueron tan intensos que han dejado una profunda huella en la psique de los informáticos, con el resultado de que hasta ahora todos los libros sobre informática han intentado enseñar a sus desconcertados lectores la forma de realizar esta incomprensible tarea. Sin embargo, para el trabajo cotidiano en informática es tan poco necesario conocer el cálculo binario, como para un pasajero de avión saber sobre las leyes de la astronavegación.

Los programadores en lenguaje máquina se dieron pronto cuenta de que estaban repitiendo trozos de código, por lo que inventaron las subrutinas (véase p. 82) y las “macro-instrucciones”. Una macro, como se las acostumbra a denominar, es una parte de un código que se utiliza como equivalente a una sentencia única. Si tenemos un trozo de código para sumar dos números decimales, lo insertaremos como una macro cada vez que se tenga que realizar esta operación. Al cabo de un tiempo dejaremos de verlo como un código: será simplemente la instrucción “sumar” (ADD). Con el tiempo habremos escrito todas las funciones ordinarias y tendremos una biblioteca de macros que podrá ponerse en acción escribiendo una lista de sus nombres como “programa”:

ENTER NUMERO
ADD NUMERO, 6, RESPUESTA
PRINT RESPUESTA

Así es como lo veríamos, llamando a las macros “ENTER”, “ADD” y “PRINT”. Las variables que necesitan —NUMERO en ENTER; NUMERO, 6 y RESPUESTA en ADD; RESPUESTA en PRINT— se colocan en posiciones estandarizadas y se conoce comercialmente con el nombre de “argumentos”.

Este no es todavía un lenguaje de alto nivel, pero progresa en esta dirección, y cualquiera que escriba mucho código en lenguaje máquina se acostumbrará a pensar los programas en dos niveles distintos: líneas de verdadero código en lenguaje máquina, en las que está trabajando, y grandes partes de código (que se utilizan como unidades) que funcionan por sí solas y no es necesario tocarlas.

Pensar los programas de este modo condujo a la división del trabajo en dos etapas. Primero, la verdadera codificación o “montaje” de las instrucciones, y luego el “ensamblaje” de las partes entre sí para construir un programa. A menudo, el ensamblador o el montador pasan por la biblioteca a buscar sus macros; en las grandes máquinas no es extraño encontrar macros escritas mucho tiempo atrás por otras manos. Este esquema se aplica también en lenguajes tradicionales tales como Assembler, Fortran, Algol y COBOL.

Originalmente el Algol no era un lenguaje de computador. Era una forma estandarizada de escribir fórmulas matemáticas para que no existiera ninguna duda sobre qué función actuaba sobre qué variable. Más tarde, alguien se dio cuenta de que, si podían escribirse macros que correspondiesen a las instrucciones matemáticas en Algol y si, además, todas las macros funcionaban de modo compatible, se podía obtener un lenguaje, a condición de que se tuviera algún trozo de programa que pudiera leer los nombres de las macros que se precisaban, sacarlas de la biblioteca y ponerlas a trabajar sobre los datos. Este programa se denomina “compilador”. Una de las cosas más hermosas de este esquema era que las diferencias existentes entre las diversas máquinas, cuyo número aumentaba rápidamente (aunque no tanto como en la actualidad), podrían camuflarse en el compilador.

Este punto de vista se introdujo a la fuerza entre los diseñadores europeos de lenguajes, debido a la aparición de muchos fabricantes de pequeños computadores que producían hardwares incompatibles, cuyas diferencias debían suavizarse y lograr, mediante el propio lenguaje, que pareciesen el mismo computador. En Estados Unidos no ocurría lo mismo. Allí IBM dominaba la escena y todos los computadores eran iguales.

En la década de los cincuenta aparecieron varios programas, llamados genéricamente “autocódigos”, para escribir en código de lenguaje máquina. A partir de estos lenguajes surgieron dos: Fortran y PL/I. El Fortran *(Formula Translation;* Traductor de Fórmulas) funcionó muy bien y continúa utilizándose en la actualidad. El PL/I intentaba ser la gran respuesta a todos los problemas del lenguaje. Sin embargo, al ser construido por partes por ingenieros, en lugar de diseñarse como un todo arquitectónico, jamás ha alcanzado popularidad.

El COBOL se desarrolló a partir de los autocódigos para ser utilizado en las empresas. Lo inventó el capitán Grace Hopper de la armada de Estados Unidos en los años cincuenta y desde aquellos días no parece que haya gustado verdaderamente a nadie. Los diseñadores del COBOL trataron de hacer un programa que pudiera entender el inglés. Poco tiempo después se abandonó esta aspiración y se intentó que el programa pudiera ser entendido por personas que comprendieran el inglés. Lo que resultó fue algo parecido a esto:

```
000480 IF CRT-STOCK CODE-SPACE GO TO END-IT
000490 IF CRT-UNIT-SIZE NOT NUMERIC GO TO CORRECT
    ERROR
000500 MOVE CRT-PROD-DESC TO PRODUCT-DESC
```

El Algol mejoró el lenguaje Fortran al permitir subrutinas más flexibles, que podían tener sus propias variables internas accesibles sólo para ellas. Sin embargo, el comité que lo diseñó estaba compuesto por personas demasiado puristas que se negaron a prestar apoyo para la realización de proyectos que permitiesen al programa Algol la obtención de datos externos con los que trabajar o comunicar sus resultados directamente a los usuarios. Concibieron un programa que se desarrollaba totalmente dentro de la máquina, sin ninguna referencia al mundo exterior. Esta desafortunada omisión impidió de alguna manera el desarrollo del Algol.

Sin embargo, aunque el Algol nunca consiguió grandes éxitos en relación al número de programas en los que se usó, supuso un gran paso teórico hacia delante. Permitió que el programador dispusiese del concepto matemático de "repetición". En Fortran una subrutina puede llamar o utilizar otra, pero no puede llamarse a sí misma. El Algol eliminó esta restricción, de manera que podemos establecer una subrutina a través de un montón de datos y hacer que se llame a sí misma tantas veces como se desee. La repetición cumple en relación a la subrutina la misma función que ésta en relación con las instrucciones individuales: permite aplicarla un número indefinido de veces.

El siguiente paso hacia delante fue el BASIC. El problema con los lenguajes compilados radica en que el compilador, primero, y el ensamblador, después, tardan mucho tiempo en procesar el programa que hemos escrito. Si, como ocurre casi siempre, el programa no funciona, debemos volver al código original, cambiarlo y compilarlo de nuevo. La duración de todas estas operaciones quizá resulte muy descorazonadora para el principiante. Con el BASIC se intenta la interpretación y la ejecución de cada línea a medida que se avanza, lo que sin duda resulta muy cómodo para el usuario ya que puede escribir un programa, ejecutarlo, ver dónde se ha equivocado, y modificarlo. Estas posibilidades han hecho del BASIC el lenguaje más popular del mundo, si tenemos en cuenta el número de máquinas que lo utilizan. Sin embargo, no resulta demasiado adecuado para el compilador, porque pasará la mayor parte del tiempo de su trabajo diario interpretando (razón por la que este software se denomina "interpretador") lo que ha propuesto al usuario a modo de sentencias de programación, y tan sólo una pequeña parte del tiempo de que dispone ejecutándolo.

Debido a que un interpretador trabaja tanto tiempo para resolver lo que se le presenta, su funcionamiento es muy lento. De hecho, hay una versión compilada de BASIC que se ejecuta diez veces más rápido de lo que se interpreta; pero, haciendo lo mismo con Assembler, la velocidad sería aún cinco veces mayor.

El Pascal, diseñado por Niklaus Wirth, trata de combinar los mejores rasgos de Algol y BASIC. En Pascal tienen que declararse todas las variables (en lugar de constituirlas a medida que se avanza, como puede hacerse con BASIC) y, al ejecutar el programa, el sistema realiza gran número de controles de las variables. Además, las subrutinas son mucho

más tratables que en BASIC, y una subrutina puede llamarse a sí misma repetidamente, cosa que no es posible en la mayoría de versiones que se utilizan del BASIC.

Cuando estos lenguajes de “alto nivel” ya llevaban cierto tiempo en circulación, la gente se dio cuenta de que todavía debían pasar muchas horas escribiendo códigos en lenguaje máquina, *assembler* y otras cosas parecidas que les producían grandes quebraderos de cabeza. Todas estas tareas tenían que hacerse porque los programas de alto nivel no compilaban lo suficiente para que fuera factible su introducción en la memoria disponible o, una vez allí, no se ejecutaban con la rapidez suficiente. Para acelerar las operaciones, se inventaron lenguajes que eran códigos máquina mejorados. Aunque eran más difíciles de escribir que los lenguajes corrientes de alto nivel, conseguían ejecutarse más rápidamente y compilaban en menos espacio. Uno de estos lenguajes es Forth, extraño ingenio que insiste, por ejemplo, en que no debemos escribir ‘3 + 4’ sino ‘3 4 +’, tal como se hacía en las primeras calculadoras.

El “C” es otro lenguaje que proporciona velocidad y condensación a expensas de la simplicidad. Funciona de manera muy parecida al Pascal excepto en que tiene menos garantías y acceso directo a cosas tales como los registros del procesador. “C” aparece como sigue:

```
compare (p1, p2, n)
    char ★p1, ★p2;
    int n; {
      register int m;
      for (m = 0: m < n && ★p1==★p2 &&
        ★p1!=’\0’; ++p1, ++p2, ++m);
      return (m==n || ★p2==★p1);}
```

lo que no parece demasiado fluido ni romántico, pero que es de gran utilidad.

Sin embargo, existe otro mundo en informática. Las personas que trabajan en la inteligencia artificial están haciendo realmente algo muy distinto de lo que hacían los matemáticos e ingenieros que inventaron los lenguajes clásicos. En la inteligencia artificial se empieza con una sentencia, por ejemplo en castellano: «Coja un cuchillo». Se sabe lo que se

quiere pero no cómo hacerlo. ¿Cómo actúan el cerebro y el ojo humanos para "ver" un cuchillo y cogerlo? ¿Qué cuchillo, en todo caso?

Para hacer frente a estos problemas se inventaron los lenguajes de "procesamiento de listas", el primero de los cuales fue Lisp. En Lisp se empieza con una grandiosa sentencia global como:

SIGNIFICADO DEL UNIVERSO (DIOSES, HOMBRES)

y a continuación se trata de ampliar con algunas sentencias subsidiarias como:

NATURALEZA (HOMBRES, MORTAL)
NATURALEZA (DIOSES, INMORTAL)

esperando llegar, a su debido tiempo, a un programa que resuelva el Significado del Universo mediante sentencias sobre MORTAL, INMORTAL y cualquier otra cosa que pueda ocurrírsenos y que sea relevante. Se confía en que eventualmente se irá descendiendo poco a poco hasta llegar a algo que el lenguaje conoce y que está escrito en código máquina. Si tenemos éxito, se habrá conseguido un programa de trabajo, en caso contrario, no.

Esta capacidad para elaborar verbos propios parece a primera vista una gran mejora sobre la rigidez de los lenguajes Von Neumann. Sin embargo, la gran flexibilidad de los lenguajes de procesamiento de listas deja muy pronto al programador agobiado por la gran masa de verbos que ha inventado y, casi con la misma rapidez, olvidado. Los lenguajes convencionales tienen tan sólo una cantidad limitada de verbos definidos con claridad en sus manuales. Y, aunque sean menos excitantes, resultan mucho más prácticos: razón, probablemente, por la cual los lenguajes de procesamiento de listas no han logrado gran popularidad en el campo del proceso de datos comercial.

En la actualidad está surgiendo, ante nuestros propios ojos, una tercera tendencia, estimulada por el gran número de personas que podrían entrar en el mundo de la informática si pudiera hacerse inteligible. Por esta razón, existe una frenética actividad, animada por enormes presiones comerciales, para encontrar formas de presentar las operaciones informáticas que permitan entenderlas sin tener que esforzarse, aparentemente,

en aprender algo nuevo. Este trabajo fue iniciado por Xerox con su máquina Star, y vio la luz pública con Lisa de Apple.

LENGUAJE MÁQUINA Y ESTRUCTURA DE DATOS

A estas alturas ya estamos en condiciones de comprender que un lenguaje de alto nivel es una herramienta para organizar e introducir en el procesador trozos de códigos en lenguaje máquina previamente escritos. Para poder enfrentarnos con el código en lenguaje máquina, necesitamos prestar mayor atención al procesador de la que pusimos en las páginas 17-20. Un verdadero procesador tiene varios registros para almacenar los datos para su manipulación. Tiene muchas más instrucciones de las tres que señalamos en aquella primera descripción, aunque en realidad no hace muchas más cosas.

Para fijar ideas, consideremos el procesador Z80 que está instalado en muchos más tipos de computador que cualquier otro. El Z80 tiene dos juegos completos de registros (de hecho, es un procesador dual) que el programador puede conmutar a placer de modo parecido como una persona que hace media puede mantener en movimiento dos agujas al mismo tiempo. El Z80 tiene en cada juego un registrador especial de 1 byte, llamado acumulador o A. Tiene tres registros de 2 bytes y un cuarto registro para guardar direcciones. Tiene, también, un segundo registro especial, F (inicial de *flags;* banderas).

Un flag es una noción específica de la informática que no tiene paralelo fuera de este campo. Es una señal que indica si algo ha ocurrido o no. Por ejemplo, nos interesa comparar un byte con otro para ver si son idénticos; en este caso, el Z80 tiene una instrucción especial para hacerlo: “CP B” compara el byte en B con el byte en A. Si son iguales, el flag cero (el segundo bit en el byte del flag) se sitúa en 0. Otras instrucciones, tales como “salto” *(jump),* también pueden utilizar el flag, de manera que el programador puede saltar a otro trozo de programa si los bytes en A y en B son iguales, o bien volver hacia atrás, cargar otro byte en B e intentarlo de nuevo, si resultan distintos.

El conjunto de instrucciones del Z80, además de las funciones de sumar, restar y comparar, que están en el núcleo del microcomputador, podrá también:

cargar bytes de la memoria a los registros y viceversa;

intercambiar los contenidos de los registros;

copiar datos de una parte de la memoria a otra;

buscar un byte particular dentro de un bloque de memoria;

hacer las funciones lógicas AND, OR, EXCLUSIVE OR (véanse pp. 25-27) en dos bytes en los registros;

llevar a cabo funciones bastante especiales que permiten hacer cálculos decimales con instrumentos binarios;

saltar de una línea del programa a otra. Estos saltos pueden controlarse mediante los contenidos de algunos de los registros. De esta manera se puede, por ejemplo, buscar la letra “a” en un área de memoria con una instrucción: si se la encuentra, se hace una cosa; si no, se hace otra;

desplazar los bits en un registro hacia la derecha o hacia la izquierda, comprobar sus valores individualmente, fijarlos o borrarlos (“fijar” un bit significa hacerlo T; borrarlo, hacerlo ‘0’;

obtener bytes de una puerta de entrada y enviarlos a una puerta de salida;

llamar subrutinas y retornar desde ellas;

interrumpir la ejecución del programa en curso y saltar a otra línea.

Si damos una ojeada a esta lista veremos pocas cosas que parezcan útiles o incluso comprensibles. La verdad es que la programación a nivel del lenguaje assembler es muy lenta y laboriosa. Se necesitaría todo un libro para proporcionar una idea adecuada de esta materia, de manera que aquí sólo podremos examinarlo superficialmente. Sin embargo, vale la pena hacerlo por la confianza y experiencia que proporciona trabajar al nivel más fundamental de la máquina.

Assembler

Tal como vimos en las páginas 31-35, el programa se encuentra en una parte de la memoria, mientras que los datos con los que se trabaja se encuentran en otra. Vamos a escribir ahora un pequeño programa para buscar la palabra “Pedro” en la memoria. Podría interesarnos hacerlo

como parte de una operación de procesamiento de textos; por ejemplo, para cambiar “Pedro” por “Juan” en el último documento que escribimos. Primero debemos encontrar “Pedro”.

Existen distintas formas de realizar este trabajo, pero la más sencilla es empezar en el extremo izquierdo de la parte de la memoria donde debemos buscar, y continuar buscando una “P” Si no se encuentra, se comprueba la próxima letra; si se encuentra, la próxima letra se compara con “e”. Si encaja, se busca la “d”; si no se encuentra, el procesador vuelve a la “P”.

La mayoría de las órdenes concretas que reconoce el Z80 consisten en un byte de instrucción seguido por dos bytes de dirección. “Saltar a la posición de memoria m n” es “11000011(m)(n)”. Incluso al programador profesional más estricto le resulta imposible recordar cien instrucciones como ésta; por esto el lenguaje en código máquina está escrito en realidad en assembler, especie de lenguaje en el que la mnemotécnica sustituye a los bytes. En el anterior ejemplo, en inglés, *Jump to m n* se convierte en ‘JP mn’.

El assembler permite también que el programador ponga nombres a posiciones de memoria particulares. Esta posibilidad no sólo ayuda al programador en su trabajo, sino que también permite escribir el código sin saber en qué lugar preciso de la memoria finalizará. La mayoría de los programas en assembler están escritos sin especificar ninguna dirección particular de memoria. Se presentan en “formato reubicable”, lo que significa que pueden cargarse en la memoria en cualquier lugar y funcionar. Esto es muy útil debido a que un programa completo acostumbra a estar construido por muchos segmentos de código en lenguaje máquina, cada uno de los cuales se escribió y comprobó por separado. Sus posiciones en la memoria para la comprobación y sus posiciones cuando finalmente se ejecutan pueden ser bastante distintas.

Estos segmentos se pegan entre sí mediante el ensamblador, programa que une trozos de código en un todo perfecto, ajustando las posiciones de memoria sobre la marcha. Ahora, después de haber establecido estos preliminares, vamos a buscar “Pedro”. Supondremos que “Pedro” está almacenado en un área de memoria cuyo primer byte se llama “NOMBRE”, y que el texto que buscamos se encuentra en un área llamada “TEXTO”.

Ejemplo de código en lenguaje máquina

DSEG			
NOMBRE:	DB	“Pedro/”	
TEXTO:	DB	“Enviaremos alguien a París - probablemente a Pedro /.”	
CSEG			
START:	LD	HL,TEXTO	
LO	LD	DE,NOMBRE	
	LD	A,(DE)	
	LD	B,A	
Ll:	LD	A,(HL)	
	CP	‘/’	;busca el final del texto
	JR	Z,NOENC	;salta si final
	CP	B	;busca el próximo carácter
	INC	HL	
	JR	NZ,L1	
	PUSH	HL	
L2:	INC	DE	
	LD	A,(DE)	
	CP	‘/’	
	JR	Z,ENCONTRADO	
	CP	(HL)	
	INC	HL	
	JR	Z,L2	
	POP	HL	
	JR	L3	
NOENC:			;el programa para tratar el caso de no-encontrado va aquí
ENCONTRADO:			;programa para tratar encontrado

Como podemos ver, el proceso es bastante complicado y a muy pocos les gustaría ganarse la vida escribiendo en este lenguaje. La gente sensata sólo escribe en lenguaje máquina cuando se ve obligada a hacerlo, lo que ocurre cuando se necesita un código que ocupe poco espacio y se ejecute rápidamente. Normalmente se emplea el software de lenguajes y sistemas o trozos de programas en lenguajes de alto nivel que se han de utilizar a menudo, ya que, de otra manera, retrasarían el trabajo.

Las empresas de software profesional odian el código en lenguaje máquina, porque es muy caro de escribir, muy difícil de mantener (es

decir, de modificar cuando aparecen errores) y porque normalmente debe escribirse el programa todo de nuevo si ha de ejecutarse en otra máquina o si el programador original cambia de trabajo.

Punteros

En su forma más simple, un puntero es una dirección donde se encuentra almacenado algo. Una variable en BASIC puede almacenarse perfectamente como un nombre en una lista de posibles nombres, cada uno de ellos con un puntero junto a él que señala a otro trozo de memoria donde se encuentra su valor. Los punteros se utilizan extensamente en la programación avanzada, y el lenguaje "C", utilizado actualmente en la programación más seria, trata principalmente de la manipulación de punteros. (Los '★' en la muestra de "C" en la página 103 señalan punteros para las variables que les siguen.) Se pueden tener punteros para punteros y hacerse un lío rápidamente. Pero lo bueno de los punteros es que permiten dirigir rutinas a diferentes cosas en la memoria sin tener que desplazar los datos al escenario de las operaciones. La programación con punteros es algo parecido a la caza nocturna de conejos utilizando un potente foco.

La principal desventaja que se presenta cuando utilizamos punteros para la señalización de las cosas que queremos investigar, es que la memoria —y el disco— se llena finalmente con cosas a las que ya no se señala. Para impedir que la memoria se llene de esta clase de desperdicios, los programas que utilizan punteros deben tener rutinas que vayan de un lado a otro como si fuesen basureros, controlando cada bit de memoria para ver si está señalado, y, en caso contrario, restituyéndolo a una lista de memoria libre, para que la próxima rutina lo utilice cuando quiera almacenar algo.

Pilas

La palabra "pila" *(stack)* tiene un sentido especial y muy importante en informática. Es una zona de memoria que se manipula de un modo especial. Normalmente, almacena datos en trozos de una longitud determinada (2 bytes en una máquina de 8 bits) para tratar las direcciones de

memoria. Tiene un indicador, para la próxima dirección vacía de la pila, que permanece en un registro especial en el procesador. Sin duda, un programa puede tener también su propia pila. Se "empuja" *(push)* un grupo de datos en la pila para almacenarlo y se "sacan" *(pop)* de nuevo cuando se desea. La mayoría de assemblers tiene mandos "PUSH" y "POP" para realizar estas operaciones automáticamente. Normalmente se acostumbra a reservar varios cientos de bytes de memoria para la pila en la parte superior del programa. Las pilas se utilizan preferentemente en las subrutinas: la pila coloca al final de una subrutina la posición de memoria a la que debe volver el programa.

Aritmética y coma flotante

Si escribimos una línea de programa que contiene el mensaje 'C=2+3' en la máquina, lo que queremos es que realice una operación y almacene el resultado bajo el nombre de variable 'C'. '2+3' es, para el computador, una serie de tres símbolos ASCII: '00110010', '00101011' y '00110011'. El computador tiene un trozo de programa llamado *parser* que reconoce series de caracteres que tengan la forma "número", "operador", "número". Cuando ve algo parecido a esto lo coge y lo trabaja. El operador V le dice que tome los dos números y haga un tipo particular de operación con ellos utilizando la parte apropiada del código máquina. Cada uno de los operadores '–', '•' o '/' (resta, multiplicación o división) activan partes distintas de programa que hacen operaciones diferentes.

No resulta excesivamente difícil hacer sumas como '2+3', ya que 2 y 3 pueden transformarse, a partir de sus códigos ASCII directamente en los bytes 00000010 y 00000011 y luego sumarse mediante la instrucción suma incorporada al procesador. Por otra parte, como un procesador de 8 bits tiene algunos registros dobles, pueden manejarse de este modo números hasta 65.536. También existe una instrucción para la resta. Sin embargo, como el resultado puede ser un número negativo, lo que normalmente se indica utilizando el primer bit en el primer byte para señalar + o –, los números que pueden representarse con 2 bytes quedan limitados al intervalo –32.766 a +32.767, y los lenguajes de alto nivel donde estos números aparecen con frecuencia, tienen una manera especial de manipular "enteros", números representados internamente por dos bytes.

Los números 2 y 3 son bastante fáciles y “2.345” no presenta grandes problemas, pero números mayores, como “10.078.489,56472” son harina de otro costal. Difícilmente pueden ponerse en forma de bytes. Pueden almacenarse como serie de caracteres, con cada dígito representado por su byte en código ASCII o por su valor en bits. Sin embargo, como un byte puede llegar a almacenar hasta 255 números y nosotros nos limitaríamos a conservar en él diez dígitos (los que van de 0 a 9), el método anterior significa un verdadero despilfarro de capacidad. En definitiva, tan sólo estaríamos utilizando una veinticincoava parte de ésta.

La solución encontrada tras largos años de evolución y de dura lucha con el lenguaje máquina, es la “aritmética de coma flotante”. La esencia de este sistema está en que los números se representan en formato científico: “2.345” se almacenaría como 2,345 E3. La primera parte de esta representación es la “mantisa”, mientras que la segunda es la “abscisa”. A cada una de ellas se le asigna un número fijo de bytes, por lo que un número en coma flotante consiste en un número fijo de enteros significativos, aunque la coma decimal puede ir en cualquier lugar. Por ejemplo, 2.345,678, 23,45678 y 0,00002345678 son esencialmente el mismo número de coma flotante con las comas colocadas en distintos lugares, dirigidas por los diferentes valores de la abscisa. De este modo la tarea de escribir un lenguaje de alto nivel resulta algo más fácil, ya que la cantidad de memoria que se necesita para cada número utilizado en el cálculo se conoce por adelantado, lo que no ocurriría si se permitiesen representaciones en series de caracteres. En el BASIC Microsoft, por ejemplo, el programador puede elegir entre tres formas distintas de representar números. Cada una de ellas proporciona una mayor precisión para los números: “enteros”, representados por dos bytes y que cubren el intervalo (–32.766 a +32.767); “simple precisión”, que se almacena como 4 bytes y que proporciona siete cifras decimales; y “doble precisión”, almacenada como 8 bytes y que proporciona catorce dígitos significativos. El programador necesita tener la capacidad de poder escoger cuál de ellos prefiere para ahorrar espacio en un programa grande y acelerar su ejecución. Una rutina que enlaza varios cálculos se ejecutará unas diez veces más deprisa si sus números son enteros que si son de “doble precisión”.

Los programas en lenguaje máquina para hacer cálculos aritméticos en coma flotante no dejan de ser difíciles, por lo que los programadores

contemporáneos son afortunados al poder adquirir rutinas ya confeccionadas en cualquiera de los lenguajes de alto nivel.

Hay varios trucos o —para decirlo en términos elegantes— métodos básicos que los programadores profesionales utilizan. El principiante sacará provecho de conocerlos porque se utilizan en el lenguaje que emplea y porque puede que quiera emplearlos directamente.

Identificadores

Ya vimos en la página 82 cómo se utilizan los identificadores en la programación en lenguaje máquina. En los lenguajes de alto nivel también son de gran ayuda.

A menudo puede interesar tomar nota de alguna condición particular para poder acceder a ella durante todo el programa. Por ejemplo, podríamos estar escribiendo un programa que se refiriese a una tercera persona y desear que se diga “él” o “ella” según el caso. Encontraremos cuál de los dos términos es el adecuado para empezar y estableceremos una variable identificadora para “M” o “F”. Entonces, de acuerdo con esto, todas las rutinas que se refieren a esta persona pueden adaptarse por sí mismas. Podríamos estar escribiendo un paquete multilingüe; si establece un identificador de lenguaje, podríamos conseguir que buscase los *prompts* en el archivo apropiado.

Almacenamiento de datos

Todo lo que se almacena en un computador son masas de ceros y unos que no tienen ningún significado intrínseco; en realidad no podemos asegurar que un conjunto de ceros y unos sea un programa para la tercera Guerra Mundial o los datos significativos para obtener menús de régimen. Todos los programadores se enfrentan al problema de desvirtuar el mundo exterior para representarlo en bits.

La parte que conecta el mundo exterior con el computador es el teclado. En las páginas 13-15 vimos que cada pulsación de una tecla se convierte en un byte único. Pero, aunque esto es magnífico, no nos sirve de gran ayuda en relación a unidades más humanas de información como números y palabras.

Una “palabra” es una serie de caracteres. Aunque en castellano las palabras están formadas normalmente como máximo de dieciocho letras, con las vocales y consonantes alternando entre sí para formar sílabas, no hay ninguna razón que impida la constitución de palabras formadas por cualquier tipo de carácter y número de bytes. (Recordemos que los caracteres ASCII tienen como bit más significativo a 0; la mayoría de computadores tienen un conjunto de caracteres “gráficos” que corresponden a bytes cuyo bit más significativo es 1.)

Esto significa que podemos almacenar palabras reales, como “mango”, dividir números como “34thg89/45” o incluso hacer cosas más extravagantes. Se acostumbra a almacenarlas manteniendo la serie de bytes ASCII con un byte delante cuyo valor se interpreta como la longitud de la serie: 5mango934thg89/ 4.

Los bytes de longitud están subrayados; lo importante es que cualquier programa que lee una de estas “series” de bytes mira simplemente el número de longitud y lee ese número de bytes. El siguiente byte es otro byte de longitud. Mantener la longitud representada por un byte único tiene como consecuencia que ninguna cadena puede tener una longitud superior a 255 caracteres, ya que 255 (FF Hex) es el mayor valor que puede tener un byte (véanse pp. 13-15).

En una aplicación más sofisticada, el byte de longitud puede colocarse en algún otro lugar, con un puntero en el texto real. El puntero es la dirección inicial de la serie de caracteres. Si quisiéramos cambiar “mango” por “pera” habría que mover el puntero desde la dirección de “mango” a la dirección de “pera”, modificando el byte de longitud de 5 a 4.

Después de unos pocos cambios más (de “pera” a “naranja” y luego a “ciruela”) el sistema continúa funcionando perfectamente, pero la zona donde se guardan las series de caracteres, el “espacio de las series de caracteres”, empieza a llenarse de frutas desechadas. Este problema es muy parecido al que se nos presentó al tratar del almacenamiento en discos (véase p. 65), pero aquí se resuelve de modo distinto. El recogedor de basura funciona cuando no queda ningún espacio libre en la parte superior del espacio de las series de caracteres que permita añadir una nueva fruta. Se mueve a través del espacio de las series de caracteres buscando zonas que no estén señaladas por la lista de nombres de serie de caracteres y longitudes. Entonces, desplaza todas las series de caracteres hacia abajo y reajusta los punteros. De este modo, los bits impares de

espacio libre que están en medio de la cadena reaparecen en la parte superior, libres para ser utilizados de nuevo.

Imaginemos ahora que queremos guardar una lista de posibles frutas a las que accederemos de manera más controlada que en series de caracteres separadas. Podrían ser: mango, pera, cereza, pomelo, naranja... Nos podría interesar modificar, añadir o eliminar algunas frutas de la lista. Un modo bastante simple de efectuar esta tarea es dar a cada entrada dos bytes extra que señalen a la siguiente fruta. La principal utilidad de esta operación es que si queremos cambiar de "cereza" a "limón", tan sólo tenemos que modificar los bytes que señalan hacia "cereza" de forma que señalen ahora a "limón" y a continuación hacer que "limón" señale hacia "pomelo". Esto forma lo que se denomina una "lista enlazada".

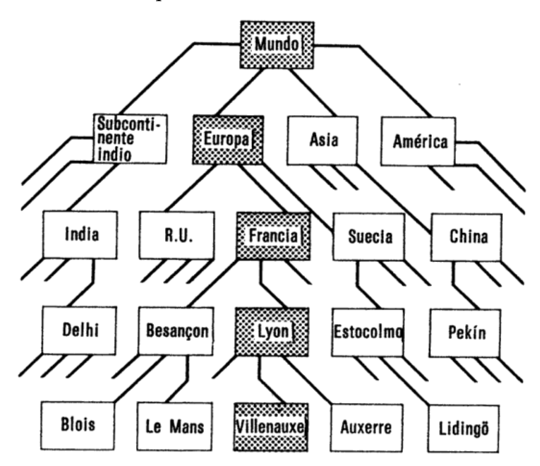


Fig. 17. Esquema para almacenar una gran lista de países y ciudades en un computador. Cada entrada (exceptuando la primera: Mundo) está señalada por otra y señala a un número variable de otras entradas. Esto hace que sea fácil de recorrer. Mundo-Europa-Francia-Lyon-Villenauxe (un pueblo cerca de París). Sin embargo, resulta difícil su aplicación a una máquina, ya que cada entrada señala a un número variable de otras entradas.

A menudo se desean hacer listas más ambiciosas con varias ramas, llamadas estructuras "arborescentes", en las cuales una elección conduce a otras. Por ejemplo, podríamos tener una lista de continentes, países y ciudades. Cada una de las conexiones en esta lista se denomina "nodo" y

señala hacia los nodos que le siguen. Así, empezamos con un nodo "mundo", que señala a los continentes: mundo a, b, c, d, ... El puntero "a" conduce a: Asia e, f, g, h... «e» conduce a: China j, k, 1, ...y «j» conduce a Pekín (figuras 17 y 18).

Esto resulta bastante fácil de almacenar estén donde estén los nodos en el "espacio de series de caracteres"; se escribe, por ejemplo, "China, j, k, 1", donde j, k, 1 son punteros de las ciudades de este país. Sin embargo, no resulta fácil programarlo, porque el número de punteros depende del número de ciudades de cada país, o del número de países en cada continente. Es más sencillo almacenar este tipo de material en un árbol cuyos nodos tienen sólo dos ramas.

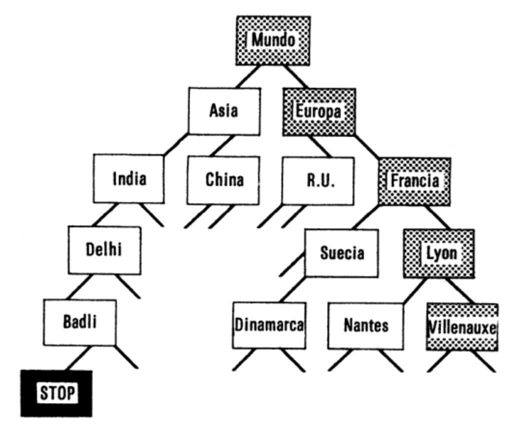


Fig. 18. Un esquema mejor es el que cada entrada señala sólo a otras dos. Cada entrada tiene direcciones de 2 bytes después de ella, que almacenan los comienzos de los próximos ítems en la lista. En la parte delantera de cada entrada está la dirección del ítem que señala hacia ella, de modo que la lista puede buscarse de abajo arriba; por ejemplo, para encontrar en que país está Villenauxe. Este esquema se denomina "arborescente".

Así, los nodos y sus punteros tienen un formato estandarizado: "China, a, b". Uno de los inconvenientes es, por ejemplo, que los continentes no aparecen todos al mismo nivel, por lo que no podemos deducir lo que es y adonde pertenece cada cosa. Una forma de tratar este problema consistiría en encadenar hacia la izquierda todas las cosas de la misma clase, con un marcador para indicar cuándo cambian. De esta forma se puede

almacenar gran cantidad de información, ya que el número de nodos se dobla en cada nivel. Un árbol entero con diez niveles de nodos almacenará 1.024 artículos; uno con veinte niveles almacenará un millón. Esto es suficiente si sólo pretendemos saber, por ejemplo, qué ciudades están en Asia; pero, si queremos volver al árbol para encontrar qué país contiene al pueblo de Villenauxe (en Francia, cerca de París), la tarea se complica bastante: el programa tiene que vagar lastimeramente preguntando: «¿Hay alguien señalando a Villenauxe?» para encontrar la ciudad de la que depende, lo que exigirá procesar una enorme cantidad de datos.

La respuesta a este problema se encuentra en la disposición otorgada a los punteros que actúan tanto hacia delante como hacia atrás. Los nodos pueden almacenar cosas muy distintas a las poblaciones y países que hemos visto. Quizá materiales utilizados en un proceso de fabricación, compañías en un sector industrial, o, en los sistemas expertos (véase p. 96), reglas para manejar información de otros nodos: «Si el paciente echa espuma por la boca, tiene fiebre y ha sido mordido por un perro enloquecido, entonces...».

Clasificación

Se ha dicho que en un momento determinado el 20% de todos los computadores del mundo están clasificando datos. En realidad, esto se dijo antes de que el Sinclair ZX81 fuese el computador más vendido del mundo: pero de todas maneras ilustra el curioso hecho de que la gente siente una pasión irracional por tener todos sus datos clasificados en algún tipo de orden. Tal como indica Wilhelm Knuth: «Los datos clasificados en orden alfabético parece que gozan a menudo de gran autoridad, incluso cuando la información numérica que se les asocia ha sido erróneamente calculada.»

Otra cosa que debemos tener en cuenta es que clasificar ocupa mucho tiempo.

“Clasificar” un archivo significa tomar los records que contiene y disponerlos en algún tipo de orden. Pueden ser records de empleados, clasificados alfabéticamente:

Alsina, José
Borja, Luis

Borja, Tomás

Fíjese que los Borja se han clasificado también por sus nombres. Podríamos clasificar a la gente según sus ingresos:

Robledo, Andrés, 8.500 dólares
Borja, Luis, 7.100 dólares
Borja, Tomás, 6.500 dólares

También podríamos hacer clasificaciones mucho más complicadas. Por ejemplo, las oficinas de correos de muchos países ofrecen al público tarifas reducidas para grandes envíos de cartas si se entregan clasificados según el código postal, ahorrándoles así trabajo. Para hacerlo se requiere un programa especialmente escrito que conozca los códigos postales.

Pero, de momento, mantengámonos en la clasificación alfabética. Tenemos cierto número de records y queremos clasificarlos alfabéticamente por la primera palabra. Existen docenas de formas distintas para realizar este trabajo. Knuth dedica 379 densas páginas a este tema, de modo que, evidentemente, no podremos hacerle justicia en este apartado. Escribe sobre la clasificación por: inserción, intercambio, selección, fusión y distribución. Al escribir un programa para clasificar, hay dos objetivos en conflicto. Por un lado, se desea que el programa se ejecute en la menor cantidad de memoria posible (o, al menos, en una cantidad de memoria que no supere la disponible). Por otro, se persigue que su ejecución dure el menor tiempo posible o, cuando no, menos tiempo que el de la vida del computador. La actividad más importante en la clasificación es comparar dos cosas entre sí, por ejemplo dos nombres —¿este nombre debería ir antes o después de este otro?— Cualquier programa de clasificación debe realizar muchas comparaciones y esto exige tiempo. El arte de la clasificación consiste, en gran parte, en la creación de algoritmos, métodos inteligentes que utilizan el menor número posible de comparaciones entre una cosa y otra.

Sólo con el propósito de hacernos una idea del problema, vamos a escribir un pequeño programa en BASIC que clasifique palabras en orden alfabético. Estas palabras se dispondrán en la memoria en una matriz de caracteres W$. (“REM” significa que el texto que le sigue es un comentario aclaratorio.)

```
1 'REM PRIMERO ENTRE ALGUNAS PALABRAS
10 DIM W$(100)
20 INPUT"Entre una palabra y pulse Return - Sólo Return para
   salir";W$(N)
30 IF W$(N)=" "THEN 50 'REM SALIDA SI NO HAY PALABRA
40 N=N+l:GOTO 20 REM N CUENTA PALABRAS
50 PRINT N;"Palabras entradas-clasificación"
60 SWP=0 'REM FLAG PARA INDICAR SI SE NECESITA OTRO
   PASO
70 FOR K=0 TO N=l
80 IF W$(K) > W$(K+1) THEN SWAP W$(K),W$(K+1):SWP=1 'REM
   VER TEXTO
90 NEXT K
100 IF SWP=1 THEN PSS=PSS+l:?"Pasada";PSS:GOTO 60 'REM
   CUENTA PASADAS, HACE OTRA SI HAY UN SWAPS ON.
110 'REM AHORA IMPRIME LOS RESULTADOS
120 FOR K=0 TO N
130 PRINT W$(K);" ";
140 NEXT K
```

Lo primero que exige este pequeño programa es que se entren algunas palabras para clasificar; podría, evidentemente, obtenerlas a partir de una ficha o de un conjunto de datos. Luego, pasa por ellas en la línea 80, comparando cada palabra —W$(K)— con la que se encuentra situada a su derecha —W$(K+1)— para ver si es “mayor”. BASIC permite usar el símbolo ‘>’ con series de caracteres; compara el valor ASCII de los caracteres primero, segundo, tercero, etc., en las dos palabras. Si alguno de ellos es mayor, entonces la comprobación sale bien. Esto produce una clasificación del tipo listín telefónico. Supongamos que comparamos “banana” con “banal”. Ambas palabras son iguales hasta que llegamos a la segunda “n” en “banana”. Esta letra es 110 en ASCII, y es mayor que 108 en ASCII (valor de “1” en “banal”); por consiguiente, el examen continuaría, y moveríamos “banal” a la izquierda de “banana”. Esto se realiza con la instrucción “SWAP” (intercambiar). Si su BASIC no posee esta instrucción, hará falta una subrutina como ésta:

```
1000 N$=W$(K)
1010 W$(K)=W$(K+1)
```

```
1020 WS(K+1)=N$
1030 RETURN
```

y se tendrá que modificar la línea 80 de la siguiente forma:

```
80 IFW$(K) > W$(K+1)THEN GOSUB 1000:SWP=1
```

Obsérvese que todas las palabras de la línea 80 deben estar en la misma caja (ALTA o baja; mayúscula o minúscula). La comprobación funcionará con “banal” o “banana” o “BANAL” y “BANANA” o “Banal” o “Banana”. Pero no dará resultado con “banal” y “BANANA”, porque las letras mayúsculas tienen todas códigos ASCII menores que sus correspondientes minúsculas (véanse pp. 13-15). Una rutina de clasificación adecuada debería poder transformar todas las palabras en mayúsculas o minúsculas antes de compararlas.

Esta es una clasificación de “burbuja” porque las palabras “flotan” hacia la izquierda a medida que la clasificación avanza. Es la clasificación más sencilla que puede escribirse; pero, como veremos al ejecutarla, es muy poco eficaz porque han de hacerse N pasadas a través de la lista, haciendo N comparaciones cada vez, de manera que su tiempo de ejecución es proporcional a N^2. Knuth dice que: «Lo único bueno de la clasificación de burbujas es su nombre pegadizo.»

El tiempo razonable para la ejecución de una clasificación debería ser proporcional a Log(N). Lo que esto significa en la práctica puede verse comparando los dos criterios:

Records	N^2	N•LOG(N)
10	100	23
100	10.000	460
1.000	1.000.000	6.908
10.000	100.000.000	9.213

Un programa de clasificación bien escrito tardará tanto tiempo en pasar por 10.000 records como una clasificación de burbujas en pasar por 100.

Hashing

Otra técnica empleada por los computadores de gran tamaño, que vale la pena conocer es el *Hashing* (trituración). El nombre sugiere un procedimiento bastante brutal y así es exactamente. La idea consiste en seleccionar cosas tales como palabras, nombres de personas, cantidades de dinero (entidades reconocidas del mundo real) y manipularlas para obtener números de índices únicos. Supongamos que estamos clasificando un montón de facturas pertenecientes a varias compañías y que queremos ponerlas de alguna forma en carpetas para encontrarlas cuando lo deseemos. Una forma de proceder sería tomar el valor ASCII de la primera letra del nombre, sumarlo al valor ASCII de la segunda dividido por diez, sumarlo al tercer valor dividido por cien y así sucesivamente:

Nombre	Hash código)
juan	119,25
fede	115,51
harrods	115,966
maceys	120,804

De este modo, a cada nombre se le asigna un número que tiene la garantía de ser único, pero que, al mismo tiempo, estará dentro de unos márgenes reducidos. Sin duda, existen millones de formas distintas de producir hashings. Precisamente, la búsqueda de métodos que produzcan valores únicos sin ocupar mucho espacio es una de las cosas más divertidas que pueden hacerse en informática.

SISTEMAS EXPERTOS

Podría parecer que para quienes se inician en la informática la dificultad estriba en escribir programas. Sin embargo, lo realmente difícil no es cómo utilizar un código poco familiar, sino cómo manipular toda la gama de conceptos nuevos que se crean en programación. Como veremos en las páginas 150 a 156 cuando hablemos de búsquedas lógicas en bases de datos, los computadores presentan dificultades intelectuales que no aparecen cuando se trabaja simplemente con lápiz y papel. Cuando se apren-

de a programar, se aprende al mismo tiempo —sin darse uno cuenta de ello— a comprender algunos de estos problemas.

Lo que el principiante necesita no es tanto que se le ayude a programar como que se le ayude a pensar. Y esto es lo que, en términos generales, intenta hacer el software llamado “sistemas expertos” o sistemas “basados en el conocimiento”.

El más sencillo de estos sistemas es un gestor de una base de datos (véase p. 109). Indexa diversos tipos de datos y permite obtener información de ellos. «El 14 de enero de 1982 vendí algo a Binks & Co. ¿Qué fue lo que les vendí?» Un sistema que sólo contesta este tipo de preguntas no puede ser considerado experto, aunque realmente sabe mucho. El nivel superior es el de los capaces de indexar y proporcionar referencias cruzadas de las informaciones que les han sido suministradas por un “oráculo” humano o por un experto y devolverlas al usuario de forma inteligente. Estos sistemas pueden habérselas con una buena dosis de incertidumbre en la visión del usuario sobre lo que está pasando. Imagínese una base de datos que le indique cómo reparar su coche. Podría tener con ella una conversación tal como la que sigue (“Y” es usted, “X”, el sistema experto):

Y: No arranca
X: ¿Hay gasolina en el depósito?
Y: Sí
X: ¿Está descargada la batería?
Y: No lo creo
X: ¿Con qué grado de seguridad?
Y: 80%
X: Revise las bujías

El sistema experto puede habérselas con respuestas inseguras. No es necesario afirmar rotundamente si la batería está o no descargada; puede darse simplemente un porcentaje. El sistema podría continuar preguntando si las bujías están bien (quizás usted estuviese seguro de este hecho, sólo en un 50%). Al final de las preguntas podría dar una lista de los posibles causantes del problema e indicarle además la probabilidad que tiene cada uno de ellos de serlo efectivamente.

Ahora bien, para realizar un programa como éste se necesita un oráculo que indique al programa embrión, que debe convertirse en exper-

to en arreglar coches, que si usted está un 80% seguro de que la batería funciona, el motor no se enciende, hay gasolina en el depósito y las bujías están bien con más de un 50% de probabilidad, es muy posible que haya algún cable suelto bajo el capó. El problema estriba en que el programa no trata realmente de la mecánica del automóvil, sino más bien de lo que piensan de ella personas no profesionales. Para hacer que un programa como el expuesto funcionase adecuadamente, sería necesario realizar una encuesta muy amplia entre propietarios de vehículos averiados que no fueran expertos en mecánica del automóvil, para averiguar qué probabilidad existe de que estén en lo cierto cuando afirman que están seguros en un 80% de que la batería está bien. Se afirma que para determinadas situaciones existen oráculos capaces de realizar este tipo de juicios.

Todavía de mayor utilidad sería un sistema capaz de aprender. Se le podrían presentar gran cantidad de hechos y pedirle que deduzca algunas reglas. Por supuesto, estos hechos deben estar de alguna manera en el computador y como los computadores no pueden revisar automóviles, ni levantar capós para ver cómo están las cosas ni comprobar si el depósito de la gasolina está lleno, es preciso introducirlos como texto o números en un disco.

Imagínese que le pide a un sistema de este tipo que examine los movimientos de stocks y personal en los dos últimos años. Tras retirarse a repasar los archivos y meditar sobre ellos, volvería y le diría: «Me parece que cuando se le acaban los botones azules durante los meses de invierno, contrata dos nuevos trabajadores.» Usted exclamaría «¡Espléndido!» y conseguiría así reducir sus gastos de plantilla ordenando una provisión de botones azules a la vuelta de sus vacaciones de verano.

Un sistema semejante, escrito por Richard Forsyth de North London Polytechnic, fue utilizado para examinar las fichas hospitalarias de pacientes con ataques de corazón. Se le pidió que examinase lo que se sabía acerca de los pacientes en el momento de su admisión y que descubriese el mejor indicador de sus posibilidades de supervivencia. Halló que si la presión arterial media de los pacientes (medida en mm de mercurio) es mayor que 61 menos el volumen de orina eliminada (en ml/h), lo más probable es que el paciente viva; en caso contrario, moriría. Esto sorprendió a los médicos que habían visto a mucha gente morir de ataques

de corazón, pero nunca se habían percatado de las correlaciones descubiertas por el computador.

En principio, un programa de este tipo podría llevar a cabo lo que se supone que hacían los “generadores de programas”. Para escribir un programa de contabilidad, no sería necesario investigar primero lo que reflejan exactamente los libros y cómo puede lograrse que el computador lo haga; bastaría con mostrarle los libros de los últimos dos años y dejarle que dedujera por sí mismo lo que en ellos se halla reflejado. Por supuesto, al computador se le podrían aclarar puntos oscuros, pero sería él quien haría la mayor parte del trabajo.

Este tipo de software está empezando a salir de los laboratorios de inteligencia artificial. Uno de estos sistemas es el *Analog Concept Learning Systems* de la Universidad de Edimburgo, que puede ser ejecutado en máquinas tan pequeñas como la Apple.

LA LEY DE ZIPF

En informática, la mayor parte de los problemas, y también el desafío y la diversión, provienen del hecho de que los computadores permiten manejar volúmenes de información mucho mayores y a mucha más velocidad que si se utiliza únicamente papel y lápiz. Pueden manejarse más cosas en mayor número de formas y así no es sorprendente que aparezcan nuevos tipos de comportamientos.

Una de las leyes más interesantes sobre el comportamiento de grandes masas de información fue descubierta en la década de los cuarenta por un sociólogo americano llamado George Zipf [4]. Aunque su trabajo es de gran importancia para un mundo informatizado, Zipf lo llevó a cabo, con inmensa laboriosidad, utilizando sólo papel y lápiz. Empezó examinando la frecuencia con que aparecen las distintas palabras en un texto en inglés. Repasó largos trozos de prosa contando el número de veces que se repetía cada palabra. Después las dispuso ordenadamente de manera que la que aparecía con más frecuencia ocupase el primer lugar, a continuación la siguiente y así sucesivamente. Entonces dispuso los resultados en un gráfico.

---

[4] G. K. Zipf, *Human Behavior and the Principie of Effort*, Addison-Wesley, 1949.

Este gráfico tiene el aspecto que podía esperarse: las palabras más raras son las menos usadas. Sin embargo, el gráfico no desciende directamente al eje horizontal porque siempre hay nuevas palabras raras que hacen que la curva se desplace más a la derecha.

Su siguiente paso fue trazar un nuevo gráfico tomando como variables el orden y el logaritmo de la frecuencia. El resultado fue realmente sorprendente. Obtuvo una línea recta. Para quienes no están versados en matemáticas, esto puede no tener demasiado interés, pero significa que la frecuencia y el orden estaban relacionados por una ecuación como ésta:

$$\log F = -k(\log O) + 1$$

donde k y 1 son constantes. Podemos considerar 1 como el log de otra constante, por ejemplo de m, de manera que la ecuación dada se escribirá ahora:

$$\log F = \log (m/O^k)$$

El valor de k resultó ser muy próximo a 1, de manera que tomando antilogaritmos:

$$F = m/O$$

lo que en lenguaje llano significa que la frecuencia con que aparecería cada palabra era proporcional a 1 dividido por el orden que ocupa. Se encontró, por ejemplo, que la palabra que ocupaba el tercer lugar en el orden de frecuencias aparecía tres veces con menos frecuencia que la más común. La centésima más común aparecía con cien veces menos frecuencia que la más común.

Todo lo que necesitamos saber ahora es: ¿Qué cantidad de común es la más común de todas las palabras? Además, para averiguarlo no necesitamos pasarnos semanas sumando palabras que aparecen en obras de Shakespeare. Si sumamos 1/1 + 1/2 + 1/3 + ... + 1/*n*, vemos que, aunque la suma crece bastante rápidamente al principio, pronto empieza a equilibrarse. Puede efectuarse con el siguiente programa:

```
10 K=1; N=0
20 N=N+l/K
30 ? K; N;" ";
```

```
40 K=K+l
50 GOTO 20
```

Mientras escribía esto ejecuté el programa en otra máquina, y obtuve los siguientes resultados:

K	N
10	2,929
100	5,187
1000	7,485
10000	9,788
100000	12,091

Al crecer K, N deja de crecer tan rápidamente y, cuando K se hace muy grande, N tiende a valer aproximadamente 12. Por supuesto, puede seguirse ejecutando el programa hasta que k valga 1 millón, 10 millones y así sucesivamente; pero se precisará mucho tiempo y, por otra parte, no descubriríamos mucho más de lo que ya sabemos.

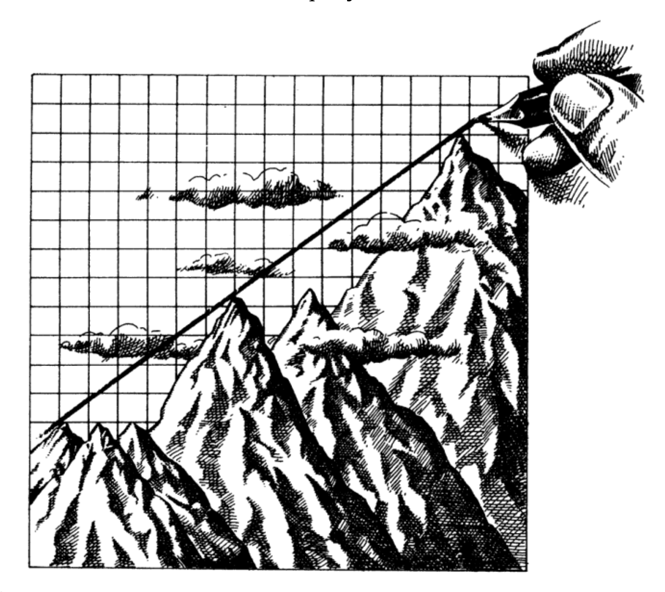


Fig. 19

Esto resulta a la vez interesante, extraño y útil. Zipf halló que la misma regla, o algo muy parecido, podía aplicarse en todo tipo de casos. Si se aplicaba al tamaño de ciudades y pueblos, resultaba que, en cualquier país, la segunda ciudad más grande era aproximadamente la mitad de la mayor, la tercera, la tercera parte, y así hasta los pueblos más pequeños. De este modo, si se conoce el número de habitantes de un país, es posible calcular cuántas poblaciones habrá entre 100 y 200.000 habitantes.

Archivos de direcciones en ciudades americanas con estructura de tipo parrilla demuestran que el número de personas que han encontrado sus esposos o esposas a dos manzanas de distancia de donde vivían era la mitad del número de personas que habían encontrado el amor en su misma manzana; una tercera parte de ese número era igual al número de personas que habían caminado tres manzanas para casarse, y así sucesivamente.

La ley de Zipf también se aplica al tamaño de las empresas; si usted conoce el volumen global de ventas anuales de microcomputadores, puede utilizar la ley de Zipf para calcular de forma aproximada la producción de las diversas compañías en la industria. La compañía más grande debería vender el doble de la siguiente, y así sucesivamente hasta la más pequeña. Si existen ya cien compañías y usted quiere empezar otra, puede calcular cuántos computadores tendría que vender cada año.

SIMULACIÓN

Una de las funciones más inútiles que pueden realizar los computadores es la de “simular” situaciones que todavía no se han producido para prever el resultado de experimentos científicos, anticipar una empresa o la economía, la influencia de sistemas de armamentos, o el curso de hipotéticas guerras; así como para pronosticar futuras tasas de interés bancario o futuras condiciones climáticas.

Por supuesto, la utilización de modelos matemáticos tiene tras de sí una tradición de siglos. Las leyes del movimiento de Newton permiten construir modelos matemáticos de sistemas tales como el constituido por el Sol, la Luna y la Tierra y comprender su movimiento. El modelo sirve para obtener una serie de ecuaciones que, una vez resueltas, dan las posiciones de los tres cuerpos en cualquier momento dado. Así, los astrónomos pueden, por ejemplo, predecir los eclipses con muchos años de anti-

cipación y (lo que es aún más útil) proporcionar a los navegantes tablas que les permitan calcular la posición de su barco o avión.

Sin embargo, en la práctica son muy pocos los sistemas que pueden escribirse mediante ecuaciones lo suficientemente simples para que resulte posible su resolución. Cualquiera que haya estudiado mecánica elemental es capaz de resolver las ecuaciones que rigen el movimiento de un proyectil disparado por un cañón en *el vacío;* son las mismas que rigen el movimiento de la Tierra y la Luna; pero nadie puede resolver fácilmente las ecuaciones que describen el movimiento del proyectil en la atmósfera, donde el aire ofrece una resistencia proporcional al cuadrado de su velocidad.

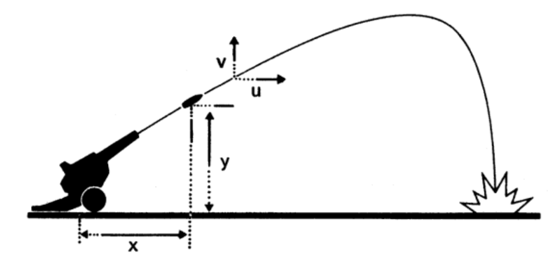


Fig. 20

La forma más fácil de descubrir cómo se mueve es utilizar la simulación mediante computador. Se divide la trayectoria del proyectil en un número elevado de pequeños saltos de corta duración: por ejemplo, de una milésima de segundo. Se considera el proyectil al inicio de un intervalo de tiempo. Se conoce su velocidad, por tanto puede calcularse la resistencia que el aire ofrecerá a su avance. Se sabe en qué medida esta resistencia frena su movimiento durante el intervalo de tiempo. Se conoce su velocidad media en el intervalo y, por tanto, a qué altura ascenderá contra la acción de la gravedad y qué distancia recorrerá en sentido horizontal. Se suman estas dos pequeñas distancias a las totales, en sentido horizontal y vertical, previamente obtenidas, y se empieza de nuevo con la posición y velocidad resultantes. Un programa simple para calcular el vuelo de un proyectil es el siguiente:

```
10 K=0,01:DT=0,1
20 1NPUT "Velocidad inicial";W
30 INPUT "Ángulo de elevación";AN
40 AN=AN•3,14/2/90
```

```
50 U=W•COS(AN):V=W•SIN(AN)
100 FOR T=0 TO 1000 STEP DT
105 PRINT X;Y
110 X=X+U•DT:Y=Y+V•DT
116 IF Y < 0 THEN STOP
120 Z=U↑2+V↑2
130 W=W–K•DT•Z
140 U=W•U/Z↑0,5
150 V=W•V/Z↑0,5–32•DT
160 NEXT T
```

La línea 10 prepara dos constantes: K determina en qué medida la resistencia del aire frena al proyectil, sin tener en cuenta la forma y el peso del proyectil, factores que de hecho también influyen; DT es el intervalo de tiempo para cada paso, que en este caso es de 0,1 segundos. Las líneas 20 y 30 piden la velocidad inicial en pies[5] por segundo y el ángulo de elevación. La línea 40 convierte los grados en radianes. La línea 50 calcula las velocidades iniciales: U en sentido horizontal, V hacia arriba.

El bucle que calcula el vuelo del proyectil en cada instante comienza en 100. (Si 1.000 no resulta lo suficientemente elevado para cubrir todo el vuelo, considérese un número más elevado). La línea 105 imprime las coordenadas horizontal (X) y vertical (Y) de la posición del proyectil. La línea 110 calcula la posición siguiente sumando a X la distancia recorrida por el proyectil en sentido horizontal (UA•DT) y a Y la recorrida en sentido vertical (V•DT).

Recuérdese que V será negativa a partir del momento en que el proyectil alcance el punto más alto de su vuelo y comience de nuevo a caer hacia el suelo, de manera que Y puede ser menor que 0, por ejemplo en el punto en que cae el proyectil. La línea 116 verifica si esto ha ocurrido y para el programa. La línea 120 calcula el cuadrado de la velocidad del proyectil en su vuelo, que, de acuerdo con el teorema de Pitágoras, es igual a la suma de los cuadrados de las velocidades en sentido horizontal y vertical. La línea 130 calcula el cambio en la velocidad W del proyectil debido a la resistencia del aire.

---

[5] El pie es una unidad de longitud anglosajona equivalente a 12 pulgadas y a 0,3048 m.

La línea 140 calcula la nueva velocidad horizontal U, multiplicando W por el cociente de dividir la velocidad horizontal primitiva por la raíz cuadrada de Z; y 150 hace lo mismo para V, con ayuda de un factor (32•DT) que se introduce para tener en cuenta la aceleración hacia abajo de la gravedad. La línea 150 calcula la nueva velocidad vertical teniendo en cuenta la resistencia del aire y la gravedad.

Los cálculos realizados en varias de las operaciones consideradas son demasiado simples para describir lo que realmente ocurre. Pero como los pasos son tan cortos, es posible ignorar complicaciones tales como la que supondría considerar la acción conjunta de la gravedad y la resistencia del aire, que nos llevaría a ecuaciones imposibles de resolver. La realización de cálculos balísticos fue una de las primeras tareas que se encomendaron a los computadores durante la segunda Guerra Mundial.

Si se ejecuta este programa, se obtiene una salida impresa de las coordenadas X e Y de la posición del proyectil. No resultaría excesivamente difícil hacer que la máquina dibujase la trayectoria del proyectil sobre la pantalla o en la impresora. Adviértase, sin embargo, hasta qué punto los resultados ofrecidos por el computador son inexactos. El método de los pequeños pasos permite que los pequeños errores introducidos en cada cálculo se sumen originando uno grande. Compruébese disparando el proyectil verticalmente hacia arriba (ángulo de elevación 90°). Debería ascender verticalmente hacia arriba y descender del mismo modo (X=0); sin embargo, aterriza a bastante distancia del origen.

La vela solar

El movimiento de una vela solar de un planeta a otro podría resultar un tema interesante para estudiarlo con un modelo de computador. En la película *Tron* aparecía una vela de este tipo, aunque es obvio que los realizadores de la película no tenían la menor idea acerca del funcionamiento de semejante objeto, ya que lo mostraban volando por encima de la Tierra, empujado por un rayo de luz rectilíneo.

La idea consiste en impulsar una nave espacial con una enorme vela de un material muy ligero y que refleje la luz. En todo momento la nave estará en órbita alrededor del Sol. El piloto de esta nave astronáutica puede dirigirla inclinando la vela diferentes ángulos en relación a los

rayos de luz. La vela tiene un área de varios kilómetros cuadrados y su superficie está azogada. La luz solar se refleja en ella y la presión de radiación (incluso los fotones tienen impulso) produce una fuerza perpendicular a la vela. Esta fuerza puede acelerar o desacelerar la nave, empujándola hacia el Sol o alejándola de él.

Si se dispone la vela de manera que la nave se mueva más rápidamente en su órbita, se alejará del Sol, superando la atracción gravitatoria. Si se inclina la vela en otra dirección, la nave se moverá más despacio y caerá hacia el interior de su órbita. Como tanto la gravedad como la presión de los rayos solares son inversamente proporcionales el cuadrado de la distancia de la nave al Sol, es posible "navegar" con igual facilidad en cualquier punto del Sistema Solar.

Modelos paso-a-paso como éste se utilizan para realizar muchos cálculos que son demasiado complicados para poderlos efectuar de una vez. Un buen ejemplo es el de la predicción meteorológica: los meteorólogos conocen (o creen conocer) las leyes físicas que rigen el comportamiento de la atmósfera y, a partir de los informes que reciben de estaciones meteorológicas de todo el mundo, tienen una idea del estado en que se encuentra la atmósfera en todo momento. Utilizando el método de cálculo paso-a-paso pueden predecir qué tiempo hará al cabo de unas pocas horas. Sin embargo, para efectuar estos cálculos con rapidez necesitan disponer de un computador lo más potente posible.

La economía es tan complicada y difícil de predecir como el tiempo, de manera que también en este caso es útil la simulación con computador, que permite combinar muchos factores que influyen en el comportamiento económico y realizar una previsión de cómo se desenvolverá la economía global de un país. Por desgracia, ninguno de los modelos hasta ahora utilizados ha tenido mucho éxito. El problema parece estribar en que la economía no está formada, a diferencia de los sistemas materiales, por cuerpos que obedecen ciegamente las leyes físicas. Por el contrario, está constituida por seres vivos e inteligentes que constantemente intentan mejorar su posición en el mercado y ajustan su comportamiento a las decisiones de sus competidores.

La economía constituye un ejemplo de una "ciencia" que parecía funcionar perfectamente mientras estuvo confiada a métodos analíticos aplicados a un número restringido de variables. La aparición de los computadores hizo posible someter a prueba predicciones sobre la economía en

su conjunto. Con lo que, por desgracia, se puso de manifiesto la magnitud de sus insuficiencias.

FRACTALES

Encontrar representaciones simplificadas del mundo real que puedan ser tratadas por computador de forma sencilla, pero que sean a la vez precisas para ser realistas, es una de las tareas más importantes de quienes trabajan con estas máquinas. Uno de los inconvenientes de los computadores es que las imágenes que con ellos se obtienen tienen las características de algo hecho por una máquina y no por la naturaleza. Los objetos naturales tienen una tosquedad y complejidad muy distintas de las formas lisas y regulares obtenidas en las pantallas e impresoras.

Por suerte, un matemático belga, Benoit Mandelbrot, dio un gran paso al presentar un método matemático simple para la descripción y la reconstrucción de las formas de montañas, costas, árboles, e incluso las finísimas circunvoluciones de las venas y arterias de nuestro cuerpo.

Mandelbrot llama a esto el estudio de “fractales”: cosas rotas o irregulares. Comienza con una pregunta que se deben haber planteado todos los estudiantes de geografía y que todos los profesores han oído más de una vez: «¿Cómo se mide una costa?» Tómese un atlas que muestre la costa este de Estados Unidos y un compás de punta fija abierto hasta la escala equivalente a 400 millas (es decir, 643,737 km en el sistema métrico decimal). Hágase “andar” al compás desde St. Stephen en la bahía de Fundy hasta Cayo Largo en Florida. La longitud de la costa resulta ser de unas 1.685 millas (2.711,742 km). Si se vuelve a realizar la misma operación con el compás más cerrado, a la escala equivalente a 100 millas (160,934 km), la longitud que se obtiene es (1.750 millas) 2.816,349 km.

Si se tiene la paciencia de repetir la operación con mapas a mayor escala y se continúa el ejercicio con el compás de punta fija abierto hasta las escalas de 50, 25, 10, 5, 1, e incluso menores, la longitud de costa que se obtendrá será cada vez mayor. Pronto se llegaría a medir en pulgadas alrededor de cada roca. Cuando la escala llegue a ser de sólo décimas de pulgada[6], la longitud de la costa aumentará de nuevo de forma impresio-

[6] Una pulgada inglesa equivale a 2,54 cm en el sistema métrico decimal.

nante, ya que se medirá siguiendo el contorno de miles de millones de guijarros. Si se reduce la escala una vez más a milésimas de pulgada, se medirán las rugosidades de la superficie de las rocas, guijarros y granos de arena. Redúzcase otra vez la escala y se entrará en la estructura cristalina de las rocas. ¿Qué se quiere decir exactamente de acuerdo con lo anterior, cuando se asigna una determinada longitud a la costa? A escala atómica la "costa" vuelve a alargarse de forma impresionante al medirse la distancia entre los átomos. Una nueva reducción de escala y se estará en el interior de los núcleos atómicos, donde la "materia" consiste casi en su totalidad en espacio vacío. La "longitud de la costa" es ahora de miles de millones de millas.

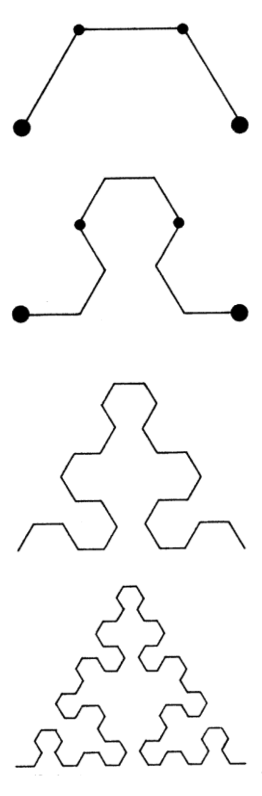


Fig. 21. Un fractal regular puede construirse empezando, por ejemplo, con una figura simple de tres lados (*arriba*). Reemplácese cada línea de la figura por una versión más pequeña de la misma (*segunda*). Repítase la operación con una versión todavía más pequeña (*tercera*) y hágase otra vez lo mismo (*cuarta*). Muy pronto todo el papel estará cubierto por algo que no es ni una línea ni una superficie, pero que tiene propiedades de ambas.

Cuando se piensa en esto, se comprende que “la línea costera” no es en realidad tal línea. Si lo fuese, tendría una longitud definida. Se asemeja más a una superficie: una especie de cinta vellosa extendida a lo largo de la costa considerada a gran escala. Pero tampoco es una superficie, porque no toda la “costa” tiene línea costera. La geometría nos enseña que una línea recta tiene una sola dimensión, una superficie tiene dos y un volumen tiene tres. Mandelbrot afirma que la costa tiene dimensión entre 1 y 2.

Podemos ilustrar lo anterior mediante un proceso repetitivo. Empiécese con una línea poligonal (fig. 21) formada por n segmentos. Hágase más pequeña multiplicando sus dimensiones por un factor adecuado y utilícese para reemplazar sus líneas rectas. Repítase una y otra vez la operación. En poco tiempo toda la superficie del papel estará cubierta por una densa masa de líneas. *Sabemos* que cada nueva versión de la forma inicial es una línea y, por tanto, el patrón obtenido es unidimensional; sin embargo, si se contempla desde una cierta distancia, se observará que, a partir de un determinado momento, no existiría ningún rincón del papel que no contenga una línea, por tanto el patrón obtenido es bidimensional. Pensamos que todo debe tener una, dos o tres dimensiones, aunque matemáticamente no es así; un número tal como 1,3 puede representar perfectamente la dimensión de un fractal. Con un computador con capacidad para dibujar gráficos, se puede escribir un programa que realice unos cuantos pasos de este proceso.

Sin embargo, con el patrón que hemos obtenido no podemos representar la línea costera de forma realista, porque, cuando lo desenmarañamos, el conjunto resulta excesivamente regular. Para superar la limitación que la regularidad supone, puede utilizarse un proceso del tipo de los denominados en estadística caminos aleatorios. Uno de los programas que casi todos los principiantes acostumbran a escribir para su computador es una versión de Drunken Duncan (Duncan el borracho). Consiste en trazar un gráfico del camino recorrido por un borracho, que partiendo de una farola, anda tambaleándose distancias aleatorias en direcciones aleatorias. Si se le concede el suficiente tiempo, habrá ido a todas partes, como un fractal o una madeja de lana.

Para conseguir algo que se parezca a una costa, hay que desenmarañar esta madeja. Un ejemplo propuesto por Mandelbrot es un juego de cara o cruz. María y Tomás lanzan la moneda por turno. Cuando cae

cara, Tomás da a María una moneda de su montón; cada vez que cae cruz, María le da una moneda a Tomás. Los gráficos que representan las unidades monetarias que cada uno de los jugadores posee en cada momento suben y bajan, y se asemejan a la sección de un terreno.

Mandelbrot ha utilizado esta idea para obtener un modelo de una costa y sus circunvoluciones. La línea costera es aleatoria, pero sigue siempre la misma dirección, y no puede tener circunvoluciones como para superponerse sobre sí misma (a diferencia de lo que ocurre con las caminatas de Drunken Duncan). Matemáticamente esto significa controlar la dimensión de la costa entre 1 y 2. De forma análoga puede obtenerse en tres dimensiones la superficie de un paisaje, y con un poco más de esfuerzo, la de un planeta.

LÁMINAS

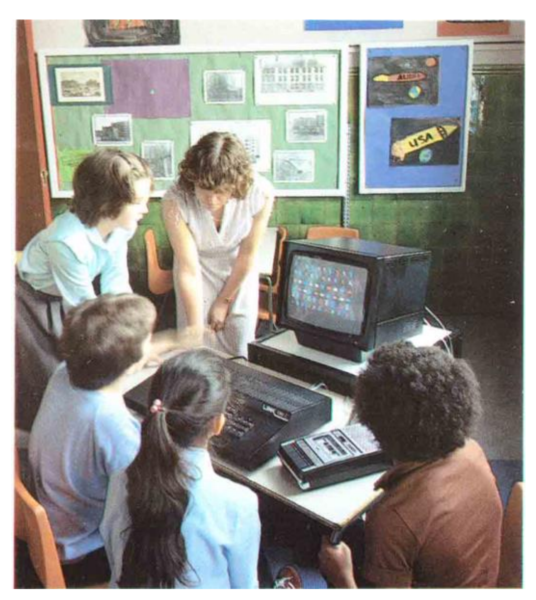


Lámina 1. Muchos gobiernos se están dando cuenta de que el futuro económico de sus países depende de la «formación informática» de la próxima generación de trabajadores. En la foto, niños de una escuela inglesa utilizan un Research Machines 480Z suministrado por el ministerio de Educación y Ciencia.

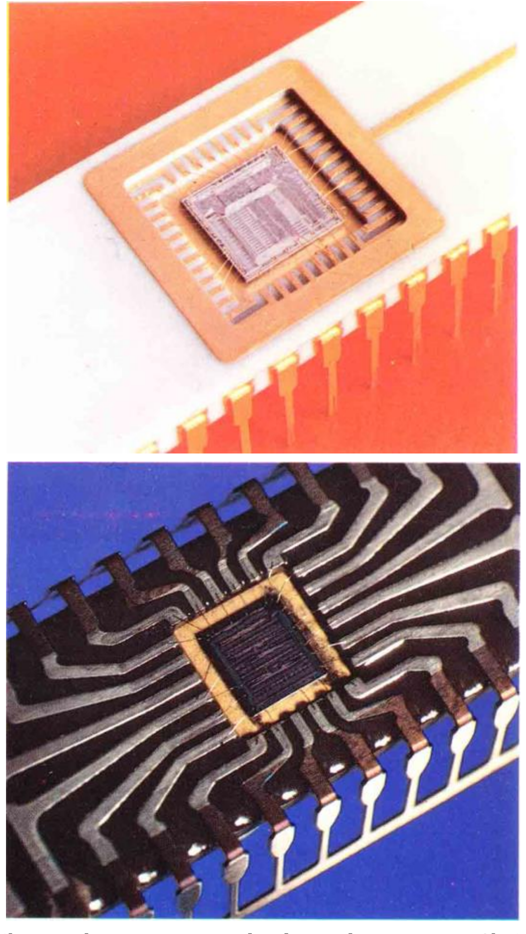


Lámina 2. *Arriba*. Un chip microprocesador dentro de su soporte. *Abajo*. Cada bloque conductor del chip está unido a cada una de las patillas de conexión externa por un alambre fino.

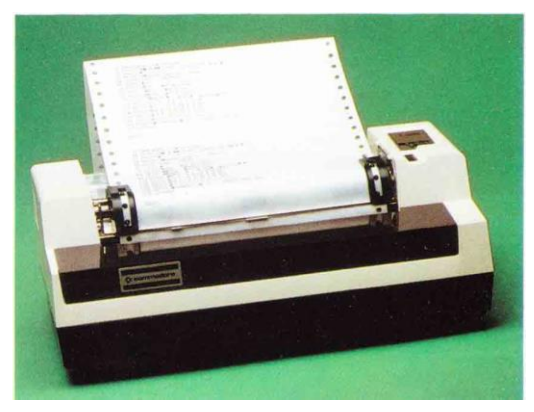


Lámina 3. Impresora de puntos convencional.

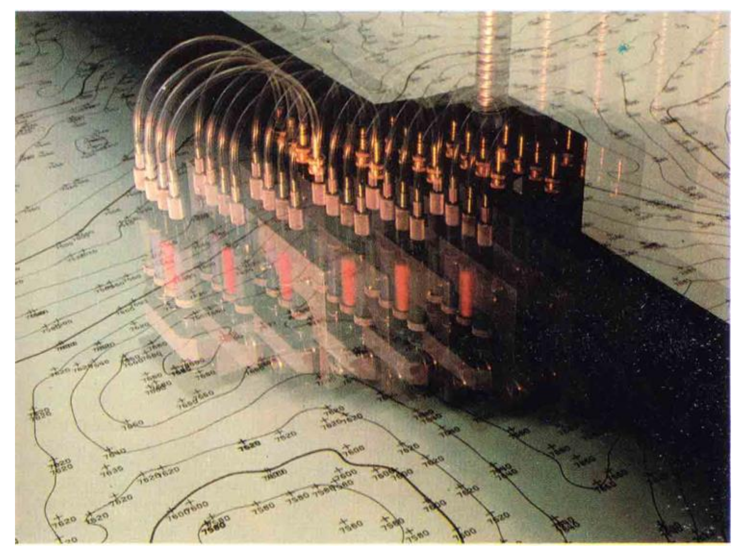


Lámina 4. Un plotter de alta velocidad dibujando curvas de nivel para un mapa.

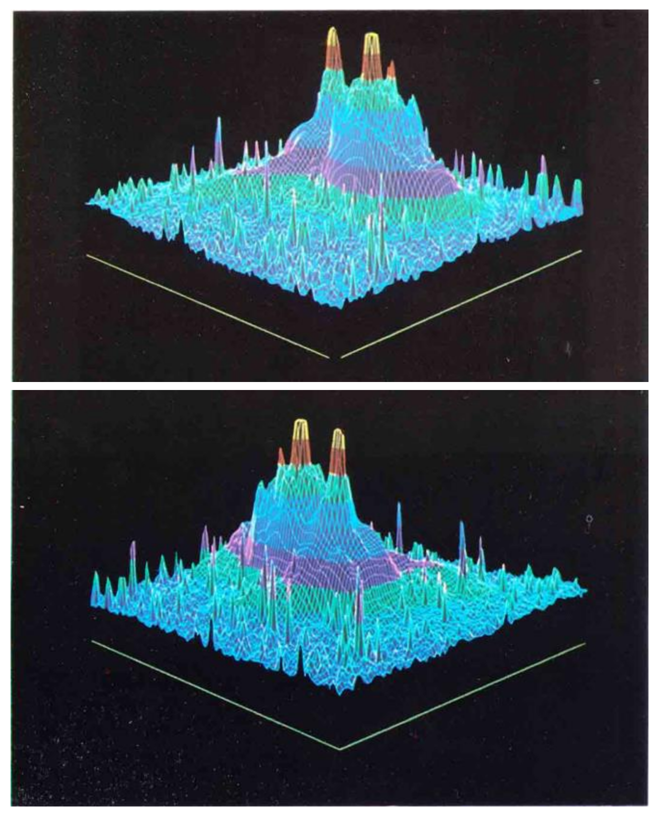


Lámina 5. Los gráficos de computadores pueden emplearse para presentar información que de otro modo resultaría difícil de digerir. Aquí se presentan dos vistas distintas de las intensidades de luz (dibujadas como montañas) en una galaxia doble M51, conocida como la nebulosa Whirlpool. Los gráficos del computador transformaron la densidad relativa de 4 millones de pixeles para cada una de las fotografías de largo tiempo de exposición en las imágenes que se muestran aquí. Se dio a cada pixel un código de color y se les estiró hacia arriba para representar así su brillo; las estrellas especiales aparecen como puntas.

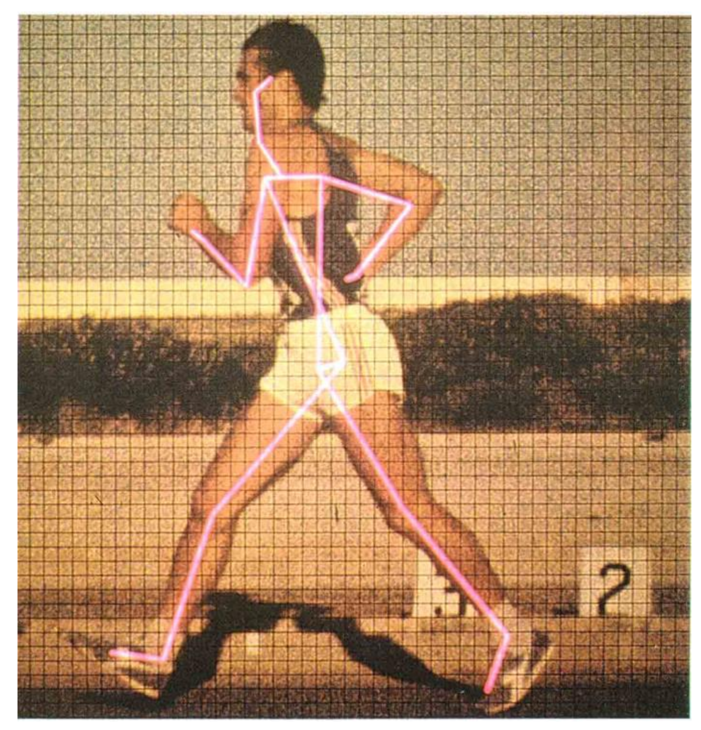


Lámina 6. Las imitaciones ergonómicas de los seres humanos contenidas en la máquina como subrutinas del programa, constituyen una parte tan esencial para el estudio del nuevo diseño como los modelos de madera de boj que sustituyen. Los científicos que estudian los movimientos de los atletas digitalizan una película de un hombre corriendo, y después trazan a mano un esquema del hombre que permite a la máquina calcular la dinámica de sus movimientos.

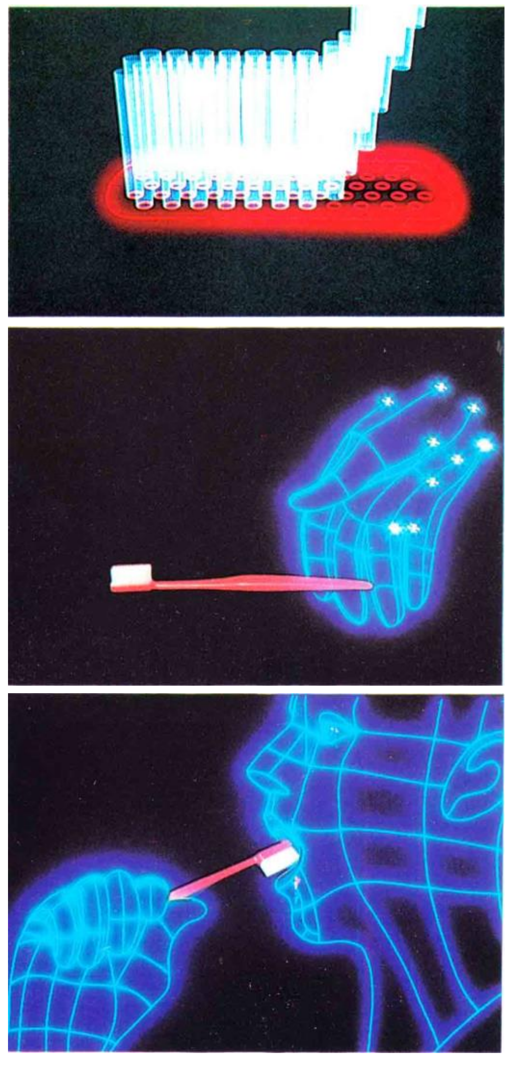
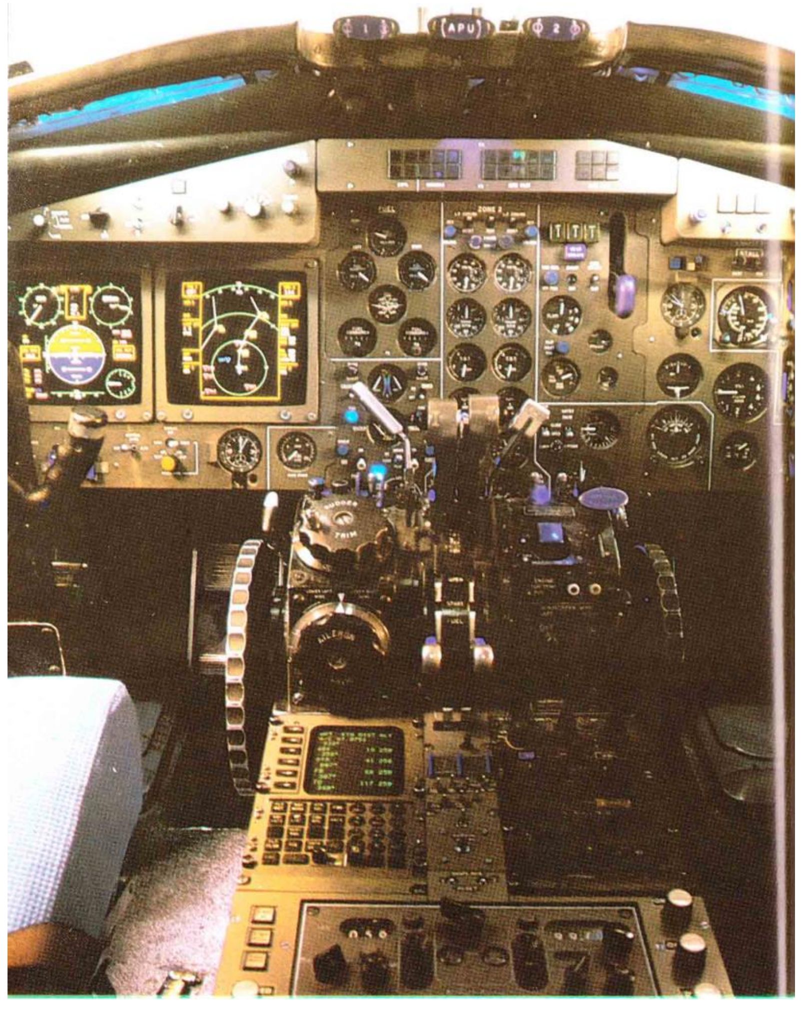


Lámina 7. *Izquierda*. Los gráficos simulados mediante computador se utilizan actualmente en los anuncios comerciales de televisión para dar a productos ordinarios un aura de alta tecnología.



Lámina 8. Los computadores entran en interacción con el mundo exterior de diversos modos y a través de muchos dispositivos diferentes. La forma del modelo se transmite a la máquina mediante un brazo digitalizador. En las articulaciones del brazo existen dispositivos muy precisos para la medición de ángulos que permiten al computador saber en todo momento la posición de la punta de contacto.



◄ Lámina 9. En los aviones se está produciendo una revolución que, aunque es prácticamente imperceptible desde el exterior, está cambiando profundamente la forma bajo la cual aquéllos son vistos por los pilotos. En esta cabina de un Aerospace 1-11 de pruebas británico pueden verse a la izquierda los nuevos visualizadores.

En lugar de los visualizadores de dial y de indicador (a la derecha), hay dos grandes pantallas en color bajo el control del computador del aparato. Obsérvese el amplio horizonte artificial en el centro del visualizador de la izquierda. Alrededor del mismo hay simulaciones de un altímetro y de un instrumento indicador de la inclinación de ascenso.

En los aviones militares, con frecuencia aparece proyectado en el parabrisas el mismo tipo de visualizador, de manera que el piloto lo ve como si estuviera frente a él en el espacio; se les conoce como *head-up display* (HUD). El HUD puede dibujar características en los paisajes externos de la tierra y el cielo que de otro modo serian invisibles, tales como indicaciones de radar de aparatos enemigos que todavía no se encuentran a la vista u objetivos camuflados en tierra.

En aviación civil se ha utilizado el mismo principio para proyectar imágenes de radar en tres dimensiones de una pista de aterrizaje situada frente al piloto pero invisible, lo que le permite aterrizar en la niebla.

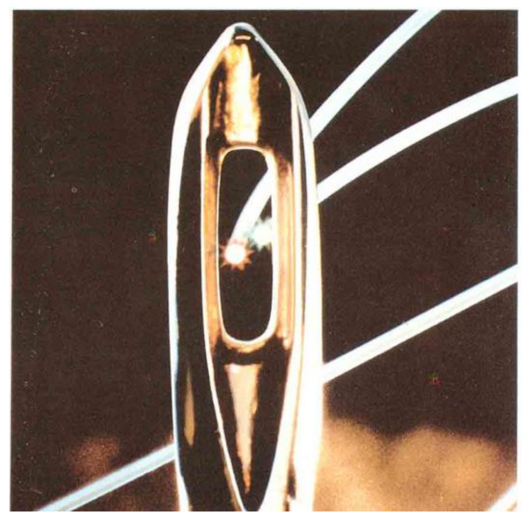


Lámina 10. Una fibra óptica: un hilo del grosor de un cabello obtenido a partir de una barra de vidrio. El vidrio tiene que ser extremadamente puro, unas mil veces más puro que el vidrio ordinario. La información se envía a través de la fibra óptica en forma de pulsaciones de rayos láser.

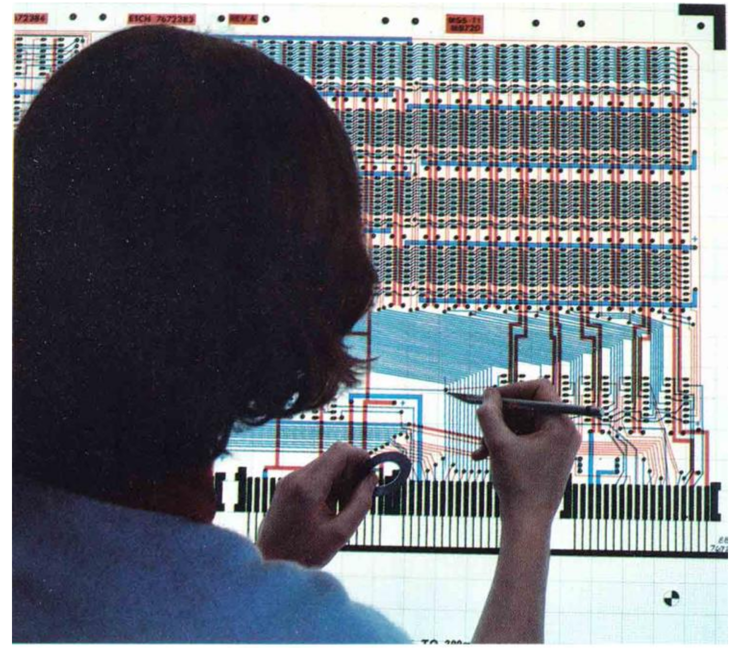


Lámina 11. Como los computadores son prácticamente ciegos, siempre representa un problema introducirles datos visuales. Aquí puede verse a una técnico que con la ayuda de un digitalizador (el anillo en su mano izquierda), realiza algunos cambios en un circuito integrado de diseño sencillo.

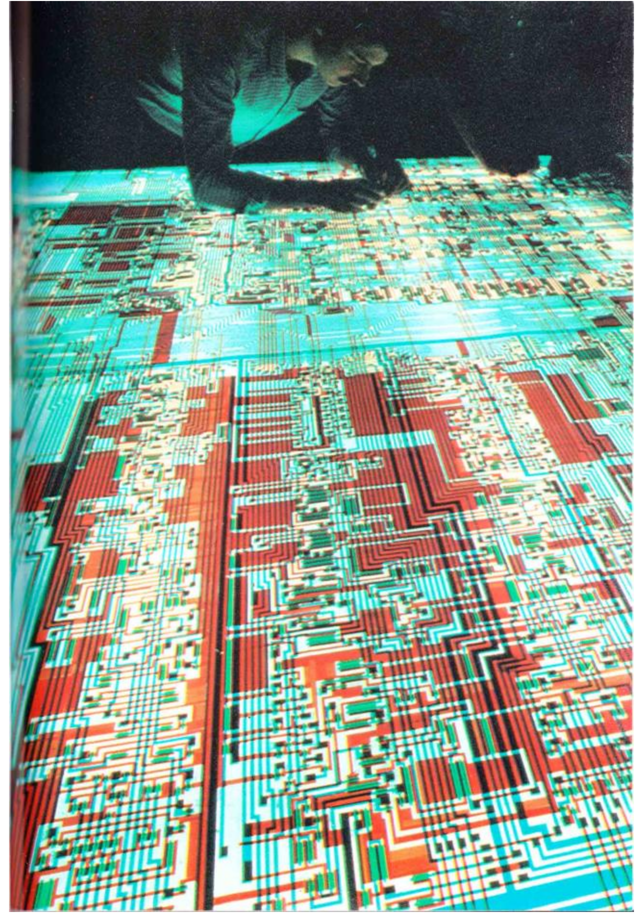


Lámina 12. Diagramas de la circuitería de un chip microprocesador, aumentados cerca de 300 veces. Cada capa del chip está representada con un color distinto, y todas las conexiones deben ser correctas antes de empezar a fabricar el chip. A medida que el ancho de línea disminuye y aumenta la complejidad de los circuitos, crece la dificultad de las tareas de diseño y control. Hoy en día ya se utilizan sistemas de diseño asistidos por computador para producir las primeras versiones de estos dibujos. El sistema CAD sólo precisa que se le diga lo que tiene que hacer el circuito y seleccionará la combinación de puertas apropiada y las colocará de la forma más eficaz.

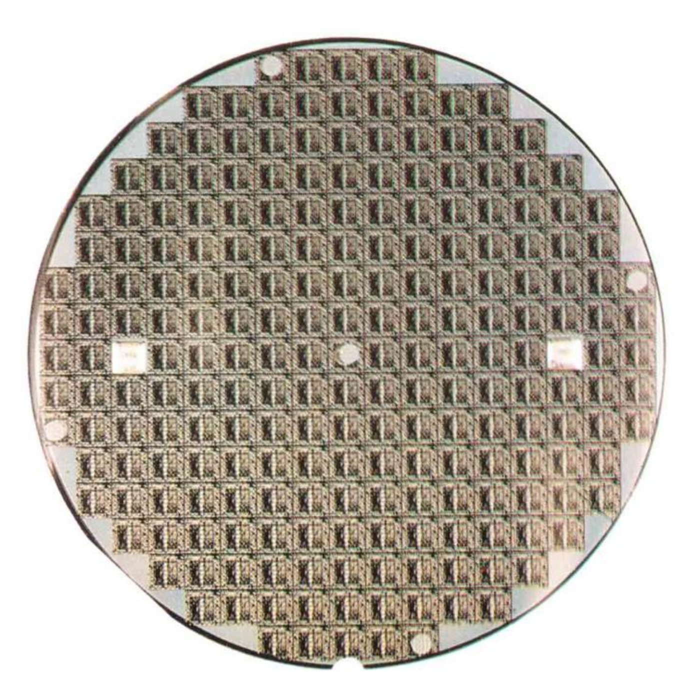


Lámina 13. A partir de finas secciones de silicio como ésta, se fabrican varios cientos de chips a la vez. Se someten a prueba directamente conectándolas con un computador y después se separan en chips. Se escogen unas cuantas posiciones de la sección para realizar las pruebas.



Lámina 14. Procesador central de un Cray 1, el computador más rápido y potente del mundo. El procesador debe ser pequeño para evitar los errores debidos a la sincronización de la velocidad de la luz. Además, genera tanto calor que debe refrigerarse con agua. Los armarios del fondo contienen memoria y computadores subordinados que manejan las entradas y salidas.

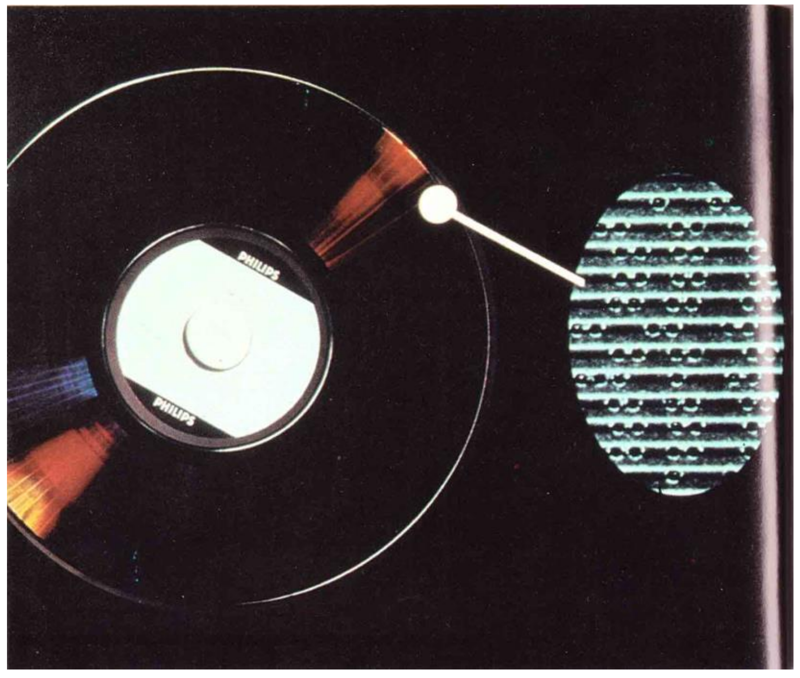


Lámina 15. Ampliación de un segmento de un disco láser, que muestra los orificios microscópicos que lee el pequeño rayo láser. Este disco puede almacenar el trabajo realizado por una mecanógrafa que trabajase continuamente durante años.

3. Informática para uso de los profesionales

SOFTWARE PARA EMPRESAS

Los paquetes de software más populares son aquellos cuyos resultados se asemejan a los distintos documentos estandarizados. Si se entra en cualquier oficina del mundo y se coge un documento cualquiera, es casi seguro que pertenecerá a una de las cuatro categorías siguientes, cada una de las cuales tiene su equivalente en informática:

Formulario - Gestor de base de datos
Lista tabulada - Gestor de base de datos
Texto - Procesador de textos
Hojas de contabilidad y presupuestos - Tipo “Visicalc”

Bases de datos

No tiene demasiado sentido disponer de un computador a menos que se tengan que realizar determinados trabajos para los que la máquina sea adecuada. Originalmente, los grandes computadores se emplearon para efectuar cálculos complicados con cantidades bastante pequeñas de datos numéricos; por ejemplo, para calcular la trayectoria de un proyectil de artillería o las condiciones en la explosión de una bomba atómica.

Al principio, los microcomputadores siguieron las huellas de sus predecesores; las primeras máquinas disponían de dispositivos de almacenamiento primitivos y resultaban adecuadas para realizar laboriosas manipulaciones de datos simples. Sin embargo, en la práctica, lo que se necesita con mayor frecuencia es realizar operaciones relativamente simples con grandes cantidades de información. Esto es, después de todo, lo que se hace en la mayoría de las oficinas. El trabajo de los oficinistas no es generalmente complicado en sí mismo, pero obliga a utilizar gran cantidad de documentos. De acuerdo con esto, la revolución que ha hecho de los microcomputadores instrumentos de la máxima utilidad en la empresa ha sido el desarrollo de dispositivos de poco costo con gran

capacidad de almacenamiento. Estos dispositivos se describen en las páginas 250-253.

Sin embargo, un sistema que sólo tuviera la virtud de permitir almacenar gran cantidad de datos no resultaría demasiado útil. Debe tener asimismo una estructura que permita encontrar con facilidad la información que se desea. Un archivador sería de poca utilidad sin secciones y cajones en los que poner los documentos y sin algún tipo de sistema para clasificar estos departamentos de manera que pueda encontrarse cualquier información archivada. Los sistemas más sofisticados de archivo de documentos tienen un índice y un registro para regular los documentos que entran y salen del archivo, para controlar quién está autorizado a examinar documentos, quién puede enmendarlos y quién no. Un sistema de almacenamiento informatizado debe poseer una estructura que realice funciones análogas a las descritas, y proporcionar esta estructura en forma utilizable.

A las masas de información estructuradas se las denomina generalmente "bases de datos". Toda base de datos posee un "gestor", que es el programa que "indexa" y clasifica la información para el usuario.

Los términos "base de datos" y "gestor de base de datos" provienen del mundo de los grandes computadores y no se ajustan del todo (por razones que expondremos más adelante) al mundo de los microcomputadores. Actualmente, los gestores de base de datos para microcomputadores con frecuencia se denominan "gestores de información" (y, probablemente, pronto se hablará de "infobases"). Los gestores de información para microcomputadores modernos tienden a concentrar más información en forma directamente reconocible por los usuarios que en las elaboradas estructuras que constituyen un gestor de base de datos convencional.

El punto de partida para proyectar cualquier base de datos es considerar que existe una gran brecha entre los archivos que gestiona el sistema operativo y la información inteligible por los usuarios. Piénsese en las cosas que se escriben en una oficina. El nombre, dirección y teléfono de un cliente, y alguna observación acerca del mismo, pueden escribirse en una ficha de forma rutinaria.

Esta información podría almacenarse en un solo archivo en el computador; pero hay dos inconvenientes: sólo podría localizarse a partir del nombre del archivo, siendo así que, en ciertas ocasiones, puede desearse

buscarla a partir del nombre del cliente, de la fecha, del número del pedido o del tipo del material que se sirvió. En segundo lugar, la mayoría de los sistemas operativos otorgan a cada archivo porciones de disco bastante grandes, con lo que reducen el número de archivos que pueden almacenarse en un disco. Después de unos cuantos recibos de entrega, sería necesario empezar un nuevo disco.

Lo que se necesita es una estructura de archivo más detallada, que permita almacenar la información que contenía la ficha de forma similar a como estaba en el papel, pero que ofrezca la posibilidad de buscar directamente cualquier cosa contenida en la ficha, superando la limitación de los sistemas tradicionales en los que es preciso buscarla por el nombre del cliente si las fichas están archivadas alfabéticamente. Existen diversos tipos que difieren en algunos detalles. El sistema que aquí se describe es el utilizado por un producto inglés llamado *Superfile*[7].

Superfile resuelve el problema de la recuperación de información almacenada, permitiendo al usuario crear un “record”. Un record es un trozo de información: un cierto número de cosas que normalmente se presentan juntas. Consta de uno o más “ítems” (unidades elementales de información). Cada ítem está formado por un “tag” (identificador) y un “valué” (valor). El tag indica qué tipo de información está en el ítem, y el valué es la información propiamente dicha. Por ejemplo, podría tenerse un record personal tal como éste:

XNAME = Tom
XNAME = Cholmondeley
SNAME = Thumb
ADD = Garden Cottage
ADD = Beanstalk Drive
ADD = Fantasyland
PAY = 50

El tag en el primer ítem es “XNAME” (una abreviatura en una sola palabra de *Christian name*, nombre de pila). Hay que distinguir entre

[7] *Superfile* ha sido publicado por mi propia compañía, Southdata Limited.

"Tom", como valor de "XNAME", y "TOM" como valor de, por ejemplo, "CATSEX" (sexo del gato) en un registro que describa gatos[8].

El registro (record) puede contener cualquier cantidad de información. Los tags pueden repetirse el número de veces que sea, del mismo modo que se repiten "XNAME" *(Christian ñame*, nombre de pila) y "ADD" (*address*, domicilio), en el ejemplo que hemos dado. No es obligado especificar la longitud de los ítems, ni es necesario decirle a Superfile antes de empezar lo que contendrá el record. Si, después de operar durante cierto tiempo el sistema de fichas informatizado, decidimos que necesitamos registrar también el número de hijos de nuestros empleados, podemos añadir el tag "CHILDNUM" *(children number*, número de hijos).

Una empresa pequeña puede necesitar un registro de clientes, de sus pedidos y de lo que se les ha cobrado por sus encargos (que puede cambiar de un cliente a otro). Un taller puede mantener una lista de las piezas de recambio, con sus descripciones, sus números de identificación, dónde se guardan y cuántas hay en almacén. Un hospital puede poseer un registro de sus pacientes, las enfermedades que padecen, las alas del hospital en que se les ingresó, los médicos que les atendieron y su dieta. Si se dispone de un gestor de base de datos, cualquiera de estas informaciones o parte de ellas será accesible directamente.

Al igual que la mayoría de los gestores de base de datos, Superfile posee *Utilities* (programas especiales) que permiten al usuario diseñar en la pantalla formularios en los que disponer la información que necesita, encontrarla, alternarla, etc., y "generadores de informes" *(reports)* para obtener extractos de información tabulados a partir de la base de datos.

Un buen paquete de formularios permite dibujar un formulario en la pantalla de modo similar a como se haría en el papel y realizar cálculos sobre la información a medida que se entra o se sale de la base de datos. Por ejemplo, si se desease escribir un programa para calcular los sueldos que perciben semanalmente los empleados de una empresa, se podría añadir un nuevo tag al record de Tom Thumb que hemos dado anteriormente, por ejemplo "HOURATE", que compute el sueldo por hora de cada uno de los empleados de la plantilla. Al final de cada semana se

---

[8] En inglés *tomcat* designa a un gato macho. *(N. del T.)*

introducirían las horas trabajadas por cada uno de ellos utilizando un formulario, que puede calcular el sueldo que corresponde a cada empleado y dar una salida impresa.

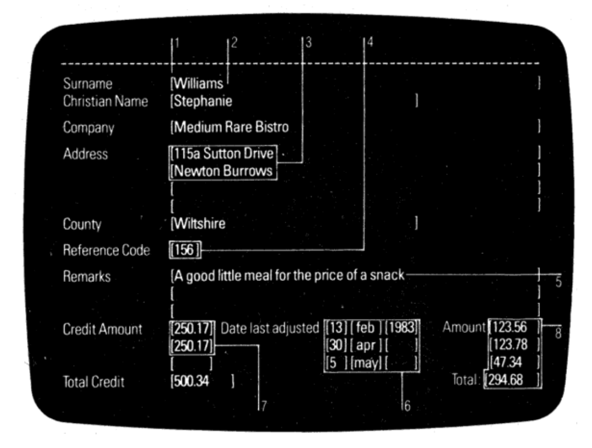


Fig. 22. Un registro (record) típico visualizado utilizando el paquete de formulaciones para la pantalla de Superfile de Southdata.

1. Los corchetes separan los “campos”, en los que se entra y visualiza la información del resto de la pantalla, donde hay *prompts* y explicaciones, al igual que en los formularios sobre papel. Es posible encontrar un registro escribiendo cualquier cosa que el usuario conozca acerca de él en cualquiera de los campos.
2. La palabra “Williams” escrita en el campo Apellido (*Súmame*) podría servir para encontrar este registro, así como cualquier otro con el mismo nombre, en la base de datos; pero también podría buscarse a partir de “suena algo así como Williams”, lo que puede resultar útil para recepcionistas.
3. Superfile permite al usuario definir un determinado número de campos como lógicamente equivalentes. Podía haberse encontrado este registro escribiendo el nombre de la ciudad, “Newton Burrows”, en cualquiera de los cuatro campos de dirección.
4. Cada registro tiene un código de referencia único.
5. Cuando se considere necesario pueden añadirse anotaciones a los registros. También podía haberse encontrado este registro buscando por la palabra “snack” entre las anotaciones.
6. Este formulario se dispuso de manera que se evitase que los usuarios entrasen por error fechas no válidas.
7. Podía haberse encontrado este formulario buscando a quienes poseen un crédito de más de 350 libras.
8. Este formulario también puede realizar operaciones aritméticas; aquí ha sumado las cantidades de dinero gastadas por la Sra. Williams.

Quizá resulte conveniente controlar la validez de la información que se ha entrado, porque la experiencia demuestra que al teclear muchos datos a menudo se cometen errores absurdos. Deberían establecerse controles que verificaran que los números son eso: números; por ejemplo, que el sueldo que corresponde a Tom no está escrito como “XXw”, lo que podría provocar dificultades en el programa. Es muy importante que todo esté escrito correctamente y para mayor garantía se aconseja que todos los datos se tecleen dos veces y acepten la información sólo después de comprobar que las entradas correspondientes son iguales en ambos casos.

El segundo programa especial de que disponen la mayoría de los gestores de base de datos es el generador de informes. Del mismo modo que los paquetes de formularios imitan el estilo de los documentos conocidos como “formularios”, el generador de informes imita a los registros tabulados. Este permite a los usuarios diseñar tipos de registros estandarizados; por ejemplo, una lista de todos los deudores de la compañía cuya deuda supere una determinada cifra o sea anterior a una fecha determinada. En un hospital podría realizarse semanalmente un registro estandarizado que mostrase cuántos “paciente-camas-día” fueron albergados en cada ala e incluyese el correspondiente número de horas trabajadas por las enfermeras, así como el coste de los equipos médicos empleados.

El generador de informes permite a los usuarios del sistema hallar cualquier cosa que deseen conocer. La decisión sobre lo que se desea conocer exactamente y acerca de la información que debe recogerse para conseguirlo, es mucho más difícil que el diseño de los formularios o los registros para hacerlo. Paradójicamente, la capacidad del computador quizá resulte a menudo agobiante porque abre posibilidades que con los sistemas tradicionales no existían. Por ejemplo, si se usasen las fichas tradicionales, el registro de la utilización de las alas del hospital mencionado precisaría de dos empleados dedicados exclusivamente a esta tarea. Sus sueldos y demás gastos que ocasionarían harían imposible el proyecto, a menos que el hospital esté totalmente seguro de que vale la pena obtener la información a ese precio. En cambio, con un sistema informatizado el coste de recoger y analizar la información es tan pequeño que las únicas limitaciones son las que marca el interés del administrador en conocer o no determinados datos y su aversión a sobrecargarse con más papeles. Como consecuencia natural del abaratamiento de la obtención de

información en sistemas de computadores de gran tamaño, a menudo se han producido verdaderas avalanchas de papel con mucha más información de la que resulta posible utilizar de forma coherente.

Bases de datos relaciónales

Un gestor de base de datos quizá resulte suficiente en situaciones en las que se desea almacenar y recuperar información. Sin embargo, como ocurre con frecuencia, en la práctica las cosas son más complicadas. Tan pronto como se empiezan a registrar transacciones comerciales, surge la primera complicación. La mayoría de las empresas hacen negocios muchas veces con la misma gente. Puede ser necesario conocer, por ejemplo, el nombre, la dirección y el teléfono de los clientes; pero, por varias razones, es interesante no verse obligado a que estas informaciones se registren cada vez que se realiza una operación con uno de ellos. Podría cometerse algún error al registrarlas, por ejemplo. Si se han vendido diez artículos al señor Rodríguez y se le ha introducido en la base de datos como “Rodríguez”, la undécima vez es posible que se produjera un error y se introdujera como “Rodrigues”. La próxima vez que lo buscase, el computador sería incapaz de encontrarlo. Si el Sr. Rodríguez ha cambiado su dirección o cualquier otro detalle, es conveniente que se pueda alterar el registro sólo en el lugar correspondiente a este detalle.

Esto significa que la base de datos está concebida para que conecte entre sí diferentes registros, de manera que el registro de una venta particular a Rodríguez se una al registro que contiene su nombre y dirección. Además, quizás interese que no se inscriban en el registro de cada transacción todos los detalles de lo que se le ha vendido. Supóngase, por ejemplo, que el 15 de enero se le vendió una docena de zapatos del tipo 46. Es mucha la información asociada a “tipo 46”: tienen la suela de madera de haya, la caña de cuero, clavos de bronce alrededor de la tira y pesan 6 kg el par. Es mejor no tener que escribir todo esto de nuevo cada vez que se vende un par. Puede conseguirse si el registro de la transacción enlaza con un registro de stocks en el que constan todas las características del “tipo 46”, el número de pares que hay en stock, cuántos están pedidos y quién los pidió. También pueden tenerse otros muchos registros acerca de los fabricantes de los zapatos que se venden.

Esto puede parecer sencillo, pero son muchos los libros que se han escrito y las reputaciones que se han consolidado dando reglas para conseguir que funcione. En el paquete Superfile del que hemos estado hablando, se conectan dos registros entre sí incluyendo el mismo ítem en cada uno de ellos. Por ejemplo, quizá sea necesario enlazar el registro que contiene la dirección de un cliente con sus transacciones. Esto puede hacerse escribiendo bajo el tag de su registro un número de identificación, por ejemplo "ID": ID=34. En cada uno de sus registros de transacciones figurará el mismo ítem, de manera que será posible encontrar las transacciones del señor Rodríguez localizando el registro de su dirección, buscando en él su "ID" y a partir de éste buscando de nuevo para localizar los registros de lo que compró. Análogamente, a partir de estos registros se podría averiguar quién fabricó los artículos que compró.

Esto es lo que entre los profesionales se conoce como una "base de datos relacional". Esta permite plantear preguntas tales como: «¿En qué ciudades se vendieron los zapatos que fabricó para nosotros la casa X?» En la práctica esto se haría examinando los registros de las fábricas para encontrar X. A continuación se averiguaría qué tipos de zapatos fabrican; después se examinarían las transacciones realizadas con los números correspondientes a estos tipos para descubrir qué clientes los compraron; por último, se examinarían los registros que contienen las direcciones de los clientes para averiguar en qué ciudades viven.

Todo esto puede resultar bastante complicado. En el caso de la fábrica de zapatos que hemos examinado es improbable que las respuestas a las sucesivas preguntas que planteamos sean muy extensas. Pero consideremos, por ejemplo, el caso de un periodista que utiliza la base de datos de compañías asociadas para investigar un fraude. Empieza con una compañía, Negocios Sucios, S.A. y pregunta: «¿Cuántas compañías tienen los mismos directores que Negocios Sucios?»

Quizá nuestro periodista espere descubrir una cadena de cargos directivos, ocupados por las mismas personas, que le conduzca a una compañía única propietaria de las demás con sede en Panamá. Mas, para encontrar esta cadena, tendrá que investigar miles de compañías y hacerlo con gran atención. Veamos más de cerca lo que puede ocurrir. Negocios Sucios tiene cinco directores: A, B, C, D y E. A también dirige Ficciones, S.A., Bajo Mano, S.A. y Manga Ancha, S.A., cada una de las cuales tiene de tres a diez directores, que son a su vez directores de varias otras

compañías. Al investigar al director A, nuestro periodista llega a la compañía Ficciones que tiene otros cuatro directores, F, G, H e I. Decide continuar investigando a F. Averigua que F es director de cinco compañías más, cualquiera de las cuales puede ser la compañía propietaria de todas las demás. El periodista deberá recordar todo lo que ha averiguado y contrastarlo con los resultados que obtenga para todos los otros directores de todas las demás compañías que tienen un director, que también sea director de Negocios Sucios. Y hasta aquí sólo ha llevado la investigación al segundo paso, imagínese las dificultades con que tropezará nuestro periodista si la compañía que busca estuviese a diez pasos de Negocios Sucios.

El ejemplo anterior demuestra que la afirmación "todo está en el computador" puede ser cierta, pero que de hecho esto no resulta de mayor utilidad que si "todo" estuviese en la Luna y se hubiesen cancelado los vuelos espaciales. Quizás estemos abocados a una "explosión informática" (de la que hablaremos en la página 192), que podría producir más información de la que los usuarios o las propias máquinas pueden utilizar.

Por otra parte, la base de datos puede acumular más información de la que se desea. Eliminar las duplicaciones de las bases de datos supone una tarea sobrecogedora. Cuando se pasa de los sistemas tradicionales de manejar información, basados en los documentos escritos en papel y donde la información permanece inalterada e inaccesible, a los sistemas basados en los computadores, en que la información se mueve muy deprisa y se automultiplica, se hace evidente que muchos procedimientos utilizados de forma rutinaria por las empresas encierran dificultades teóricas que no aparecen como tales porque la rigidez de los sistemas de manejar información basados en documentos escritos hace imposible formularlas. En cambio, a una base de datos relacional se le podría pedir que conectase a los proveedores con los clientes para anular las compras de materiales defectuosos.

Procesadores de textos

En el siguiente tipo de documento estandarizado se incluyen principalmente textos, cartas, artículos, informes, libros y memorándums. Los

paquetes de programas para producir este tipo de material se conocen como «procesadores de textos». Como la estructura de los textos, aunque complicada, es algo perfectamente comprendido, ya que los textos existen desde hace siglos, los procesadores de textos trabajan dentro de márgenes delimitados que no dejan mucho espacio a la originalidad.

Todos ellos cumplen tres funciones principales. Permiten al usuario escribir y corregir un documento en la pantalla, lo imprimen correctamente dispuesto sobre el papel y, cuando el usuario ha terminado, lo almacenan en un disco, de donde puede recuperarse cuando se desee.

Estas funciones combinadas permiten escribir en la pantalla, corregir, imprimir, realizar una nueva corrección, etc., de un documento sin tener que pasar por la agonía que supone mecanografiarlo de nuevo cada vez. Pueden almacenarse en un disco cartas estandarizadas, informes y contratos (o partes de ellos) y reproducirlos cambiando determinados elementos para obtener el documento final.

Examinemos una a una estas tres fases. La primera permite crear un texto en la pantalla. A los usuarios les interesa que sea posible escribir en el teclado como en una máquina de escribir. Necesitan poder volver atrás y cambiar partes del texto, desplazar un párrafo de arriba abajo, etc. Por ello todos los procesadores de textos permiten al usuario mover el cursor en cualquier dirección que se desee para encontrar una determinada palabra, o para cambiar una palabra (por ejemplo, «programma») por otra (en nuestro ejemplo, «programa») en todos los lugares en que aparece en el texto. Pueden acudir al disco y añadir un nuevo texto perteneciente a otro archivo, lo que les permite, por ejemplo, componer un contrato complicado escribiendo párrafos estandarizados uno detrás de otro. Los párrafos pueden estar almacenados con los nombres de las partes contratantes escritos como “N1” y “N2”; cada vez que aparecen estos símbolos en el texto, un dispositivo especial los transforma en “Dr. Jekyll” y “Mr. Hyde”, las dos partes contratantes. La próxima vez que se utilice el sistema, se podrían convertir en “Sr. Rodríguez” y “Sr. Vázquez”.

Paquetes de software más sofisticados permiten editar varios documentos a la vez; bueno, en realidad no del todo, pero los mantendrán todos al alcance permitiendo pasar de uno a otro a voluntad. Algunos sistemas dividen la pantalla en dos o más partes que se comportan como pantallas independientes en el procesamiento de textos. Se puede saltar

de una a otra, visualizar un documento en esta pantalla y mover trozos de texto entre ellas.

Cada sistema ofrece distintas posibilidades, pero las expuestas son las más corrientes. Las máquinas especialmente diseñadas para el procesamiento de textos poseen teclas y pantallas especiales; con los microcomputadores que ejecutan paquetes de procesamiento de textos se utilizan las teclas de que disponen. A menudo, las instrucciones para mover el cursor por la pantalla se dan pulsando la tecla "CONTROL" y la de una letra. Existe una máquina en la que la tecla CTRL C mueve el cursor un carácter hacia delante, mientras que la tecla CTRL L lo mueve una línea hacia delante.

Sería interesante que el texto apareciese en la pantalla exactamente como aparecerá impreso sobre el papel; en el lenguaje de algunos anuncios optimistas: «Lo que ve es lo que obtendrá». Por desgracia, esto pocas veces resulta posible ya que las mejores impresoras dan a las letras espacios proporcionales a su anchura, mientras las pantallas dan a todas ellas el mismo espacio. En otras palabras, en el papel la "i" ocupa la cuarta parte del espacio que ocupa la "m", pero en la pantalla ambas letras ocupan el mismo espacio. Además, muy pocas pantallas de computador tiene líneas de más de 80 caracteres (las más baratas tienen sólo 40), mientras que la mayoría de las impresoras escriben líneas de 132 caracteres o más. Sin embargo, a muchos fabricantes de procesadores de textos les gusta afirmar que "lo que ve es lo que obtendrá", pero por desgracia esto no es cierto.

La segunda fase del procesamiento es almacenar en un disco lo que se ha escrito en la pantalla. Esto debería hacerse de forma casi automática sin que el usuario tenga conciencia de las operaciones en el disco. En los paquetes bien escritos, cuando los usuarios mueven el cursor a través del texto, simultáneamente éste se registra o se borra en el disco, lo que les permite trabajar en documentos que pueden llegar a tener millones de caracteres.

La tercera fase del procedimiento en convertir el texto impreso lo que los usuarios han escrito en la pantalla. Como ya vimos, lo que aparece en la pantalla no será exacto a lo que se imprime en el papel.

En principio, no hay ninguna razón por la que los procesadores de textos deban limitarse a arreglar la disposición de las palabras del texto o a imprimirlo dejándolo tal como aparecía en la pantalla. En ocasiones se

desea, por ejemplo, numerar los distintos párrafos correlativamente, o imprimir el texto de manera que tenga la estructura de un poema, o simplemente escribir palabras una debajo de otra formando una lista. La máquina debe ser capaz de arreglar el texto tan bien como podría hacerlo cualquier secretaria.

La sección del programa que realiza esta tarea recibe el nombre de “formatter” (formateador). En los sistemas más antiguos y también en los main-frames, es un software especializado el que imprime los archivos de texto que crea el programa editor. Escribir estos tipos de paquetes no es en ningún modo tarea fácil, por lo que en general el público interesado tiende a comprar programas que lo hagan.

Hojas de cálculo [9]

El tipo de documento estandarizado que nos falta por considerar es la hoja de cálculo también conocida como “hoja electrónica”. La práctica de la contabilidad por partida doble se inició en Italia en el siglo XIII.

Existe actualmente un software que imita este tipo de documentos; permite disponer de filas y columnas y entrar cifras en las cuadrículas —de las ventas correspondientes al mes de marzo, por ejemplo—, después puede calcular el porcentaje que habrá que pagar en impuestos, la comisión del vendedor y los gastos generales, y proporcionar una salida impresa a pie de página del beneficio resultante. Este software es, por tanto, de gran utilidad para realizar cálculos rutinarios en relación con negocios realizados en el pasado y permite a los directores de empresa experimentar con las condiciones futuras del negocio. ¿Qué ocurre si los impuestos aumentan un 3%? Supongamos que nuestros gastos en combustible disminuyen en 14.000 dólares, pero los salarios aumentan en un 4,9%, ¿qué pasaría entonces? En vez de tener que efectuar cientos de

---

[9] En rigor este epígrafe no trata de las cuestiones contables *stricto sensu,* sino que hace referencia a la manipulación de datos, conceptos y cálculos de todo tipo en la cifra de negocios (así como a las operaciones mercantiles, financieras, bursátiles y de promoción conexas), que se reflejan en el balance (y la contabilidad) empresarial. La contabilidad en sentido estricto es objeto de estudio en el apartado siguiente. *(N. del T.)*

cálculos con una calculadora de bolsillo, puede entrarse una cifra y dejar que el software vuelva a calcular todas las demás.

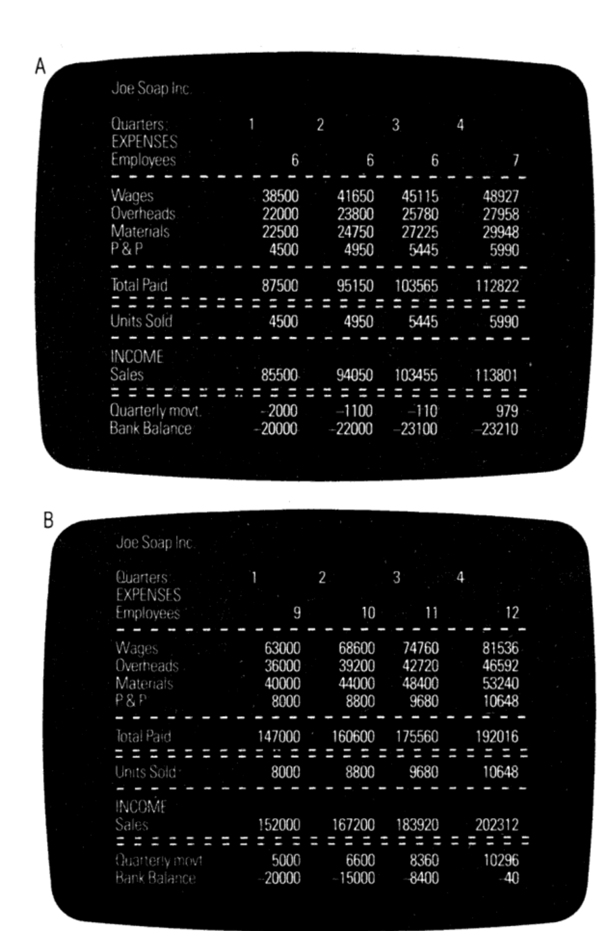
A partir de estos paquetes de software es posible desarrollar nuevos programas. Por ejemplo, si todos los banqueros desean realizar una serie de cálculos laboriosos para obtener hojas de cálculo esencialmente iguales, es posible que alguien considere que vale la pena introducir estos cálculos en el paquete de software estandarizado para producir un nuevo tipo de paquete estandarizado más especializado.

La hoja de cálculo es una manera de presentar informaciones numéricas pero no necesariamente la mejor, especialmente si lo que se busca son correlaciones entre dos grupos de cifras diferentes. A un fabricante, por ejemplo, puede interesarle descubrir el efecto sobre sus ventas de un incremento en su presupuesto de publicidad. Las cifras pueden estar todas sobre el papel, pero presentadas de manera que resulten difíciles de comprender. Por esta razón, cada día hay más gente que utiliza paquetes de software estandarizados para obtener gráficos de las cifras de su negocio. Basta simplemente con escoger dos variables (por ejemplo, los gastos en el eje vertical y el tiempo en el horizontal) y el paquete por sí solo encontrará los valores más alto y más bajo, las escalas aproximadas para el gráfico y lo dibujará en la pantalla o lo imprimirá en papel. También pueden obtenerse gráficos de dos o más cantidades conjuntamente.

Otra forma de presentar información sobre movimiento de dinero es el gráfico en forma de tarta. Tampoco en este caso supone ninguna dificultad para el software estandarizado tomar una serie de números que sumen 100 y construir con ellos un gráfico de este tipo.

Contabilidad

Como los computadores tienen una habilidad para manejar números de la que muchas personas carecen, llevar la contabilidad es una de las tareas que más comúnmente se han asignado a los microcomputadores. Una máquina no cometerá (o no debería cometer) errores aritméticos ni de procedimiento, ni olvidará lo que se le ha dicho. Bueno, esto no es en realidad del todo cierto, pero sí lo suficientemente cierto para que los paquetes de contabilidad sean muy populares.



A

Joe Soap Inc.

Quarters:	1	2	3	4
EXPENSES				
Employees	6	6	6	7
Wages	38500	41650	45115	48927
Overheads	22000	23800	25780	27958
Materials	22500	24750	27225	29948
P & P	4500	4950	5445	5990
Total Paid	87500	95150	103565	112822
Units Sold	4500	4950	5445	5990
INCOME				
Sales	85500	94050	103455	113801
Quarterly movt.	-2000	-1100	-110	979
Bank Balance	-20000	-22000	-23100	-23210

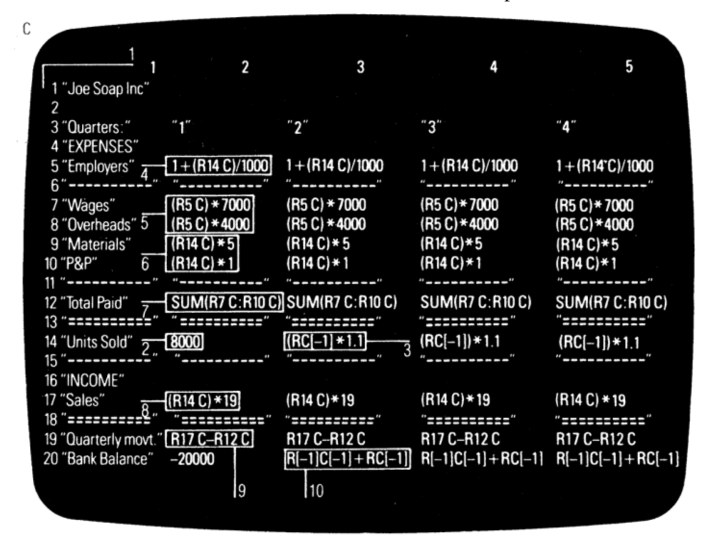
B

Joe Soap Inc.

Quarters:	1	2	3	4
EXPENSES				
Employees	9	10	11	12
Wages	63000	68600	74760	81536
Overheads	36000	39200	42720	46592
Materials	40000	44000	48400	53240
P & P	8000	8800	9680	10648
Total Paid	147000	160600	175560	192016
Units Sold	8000	8800	9680	10648
INCOME				
Sales	152000	167200	183920	202312
Quarterly movt.	5000	6600	8360	10296
Bank Balance	-20000	-15000	-8400	-40

Fig. 23. Esta hoja de cálculo muestra el plan de negocios de una empresa de estructura extremadamente simple. Se pretende conocer el número de «unidades vendidas» en el primer trimestre, y, en relación con esta cantidad, el número de empleados, el coste de los materiales y de los gastos de envío, y, por supuesto, los beneficios. La línea inferior muestra lo que está ocurriendo con la cuenta bancada de «Jabón de Joe». Se abrió con un debe de 20.000 (u.m.) [lo que le costó montar el negocio]. La hoja de cálculo le permite experimentar: si empieza vendiendo sólo 4.500 unidades, está destinado a arruinarse (Hoja A); pero si consigue vender 8.000 (Hoja B), las perspectivas son halagüeñas.

La Hoja C muestra cómo se estableció la hoja usando el multiplán de *Microsoft.*
1. Número de fila y de columna para la identificación de las «celdas».
2. Cifras iniciales de ventas. Todas las demás se calculan a partir de ellas.



3. Para los trimestres siguientes se prevé que las ventas aumenten a un ritmo de un 10% cada trimestre.
La fórmula «[RC–1]•1,1» significa: introdúzcase la cifra en esta fila *(row)* y en la columna de la izquierda, multiplicada por 1,1. La fórmula se repite en cada celda a la derecha, para producir el mismo efecto en las cifras correspondientes a los trimestres siguientes.
4. El número de empleados es 1 + el número en la fila 14 y la misma columna (unidades vendidas) dividido por 1.000; es decir, la empresa tiene un gerente y todos los demás empleados pueden manejar 1.000 unidades al mes.
5. Los costes salariales y los gastos generales han sido calculados para el número de empleados necesario para vender 7.000 y 4.000 unidades y para cada trimestre.
6. El coste de los materiales, los gastos de envío y de envasado han sido calculados a partir del número de unidades vendidas.
7. Gastos totales: suma de las filas 7 y 10 en una misma columna.
8. Los ingresos sobre la base de un precio de venta de 19 u.m. la unidad.
9. El movimiento de dinero trimestral: ingresos menos gastos.
10. El saldo bancario se abre con –20.000. Cada saldo subsiguiente es el saldo anterior más el movimiento en el trimestre anterior.

¿Qué debería hacer en principio un software de este tipo? Esencialmente, una empresa es algo muy simple. Compra cosas tales como cacahuetes u horas de trabajo de un programador y vende mantequilla de cacahuete o paquetes de software. La diferencia entre el dinero que gasta y el dinero que ingresa es lo que permite a su propietario irse de vacaciones a las Bahamas. Sería muy agradable para el propietario saber en cualquier momento si puede ir o no a las Bahamas, de manera que lo que más interesa es que la máquina mantenga un control de lo que hay en todos y cada uno de los departamentos y reste uno de otro.

Por supuesto, la contabilidad es bastante más complicada que todo esto y, simplemente, mirando por encima un libro de contabilidad es fácil darse cuenta de que a través de los siglos los contables se las han arreglado para hacer algo muy enrevesado de algo en principio muy sencillo. Este punto de vista tiene bastante de cínico, ya que en realidad incluso los asuntos financieros de una empresa pequeña pueden llegar a ser muy complicados.

La manera más sencilla de considerar una empresa es verla como una serie de contratos. Cada contrato pasa por una serie de etapas, cada una de las cuales debe ser consignada. El propietario de la empresa está obligado a pagar diversos tipos de impuestos, impuestos sobre las ventas e impuestos sobre los beneficios, que se calculan considerando el negocio desde distintos puntos de vista. Puede resultar necesario examinar todas las transacciones del propietario con el Sr. Rodríguez, qué se vendió en el último trimestre, o qué se gastó en sueldos y en publicidad.

Todo esto es en principio bastante fácil para el computador si los datos han sido almacenados en forma adecuada. Lo que hay que hacer es conservar un registro de cada contrato en la base de datos. Una vez se tiene toda la información acerca del contrato, todo lo que desee el contable puede obtenerse procesando los registros básicos. Desgraciadamente, ésta no es la manera como los contables llevan la contabilidad y se comprenderá por qué cuando se recuerde que en el pasado tenían que hacerlo sin más que lápiz y papel. En el sistema tradicional existen dos limitaciones; en primer lugar, sólo se puede buscar determinada información a partir de una característica: el nombre del proveedor si los registros están por orden alfabético, o la fecha si están por orden cronológico, pero no de ambas maneras. De ahí que en estos sistemas exista un “libro diario” y un “libro mayor”.

En segundo lugar, con los sistemas basados en los libros de contabilidad siempre pueden producirse errores en cualquier fase del proceso, lo que llevó a la práctica de la contabilidad por partida doble. En un sistema que utilice un computador nada de todo esto es necesario. El "libro diario" y el "libro mayor" son consecuencias de dos puntos de vista distintos sobre los mismos datos: buscar en la base de datos a partir de la fecha o a partir del nombre. Mientras se introduzcan los datos iniciales correctamente no es necesaria la doble entrada, porque la máquina no se equivocará en las sumas.

Por desgracia, los programadores que escriben paquetes de contabilidad empezaron con dos desventajas. No tenían bases de datos y tenían contables. Los contables querían que todo se pareciese (incluso lo que no estaba a la vista) a lo que ellos hacían. Como consecuencia de esto, los sistemas de contabilidad con computador mantienen archivos que corresponden exactamente a los diversos libros utilizados en los sistemas tradicionales, llegando incluso, en ocasiones, al ridículo que supone obligar al usuario a introducir los datos dos veces.

COMPUTADORES E IMÁGENES

La aplicación de los computadores en la obtención de imágenes y dibujos es uno de los campos de la informática más fascinantes y que más se ha extendido en los últimos años. Se desarrolló a partir de un gran número de áreas inconexas. Los computadores se utilizaban para ayudar a los delineantes de máquinas en la producción de imágenes simuladas para entrenar pilotos de aviación civil, para ayudar a los dibujantes de películas de dibujos animados y para hacer películas. También se utilizaban en la producción de efectos especiales, en el análisis de fotografías de la superficie terrestre obtenidas desde satélites, como soporte en astronomía para el análisis de imágenes de galaxias distantes, para el diseño de tejidos y para la realización de maquetas que muestran a los clientes el aspecto final que ofrecerán los proyectos de arquitectura. Los médicos los utilizaban para facilitar la interpretación de radiografías, de exploraciones *(scans)* del cerebro y de películas que mostraban el movimiento de los atletas al correr y al saltar. Ayudaban a los ergónomos a simular la interacción humana con todo tipo de máquinas fantásticas.

Siguiendo otra línea de desarrollo, grupos de especialistas intentaban construir máquinas y escribir programas que permitiesen a los computadores "ver" a través de cámaras de televisión. Lo que se pretendía era construir un artefacto que al mostrarle una habitación o una máquina fuese capaz de identificar el mobiliario o las piezas que la constituían.

Por desgracia para los aficionados no profesionales, este trabajo exige disponer de un hardware potente. Para obtener una buena representación se necesitan gráficos de alta resolución. Y los gráficos de alta resolución exigen por lo menos gran cantidad de RAM, pantallas de alta resolución y procesadores potentes. (Es posible que la próxima generación de máquinas con visión utilice las técnicas de procesamiento paralelo que se describen en las pp. 246-250.) El primer microcomputador que casi cumple estos requisitos es el Lisa de Apple; pero, en realidad, no es lo suficientemente potente ni barato como para significar la introducción de máquinas con visión en el mercado.

Dentro del amplio campo de esta tecnología pueden considerarse dos grandes grupos de aplicaciones: aquellas en que se construye una imagen en el interior del computador a partir de un plano, de un dibujo o de una fotografía (en estas aplicaciones se utiliza el computador para ayudar al artista o al delineante) y aquellas en que lo que se intenta es que la máquina "vea" en la misma forma que lo hacemos nosotros. Para demostrar que el computador realmente ha visto lo que se le ha presentado, se le puede hacer dibujar.

Dentro de estas divisiones tenemos que hacer una nueva distinción: lo que el computador imprime en la pantalla puede estar muy lejos de lo que "realmente" está haciendo. Por ejemplo, una máquina para el diseño asistido por computador *(computer-aided design*, o CAD) resulta de gran utilidad en una fábrica para diseñar una caja de cambios. El delineante utiliza la máquina para dibujar en detalle los dientes, asegurarse de que engranan perfectamente y que una parte no entorpece a la otra. Si se considera necesario, puede realizarse un análisis de las tensiones mecánicas sobre el propio diseño utilizando el computador, con lo que se logrará la seguridad de que todas las partes son lo bastante fuertes para realizar el trabajo para el cual están concebidas, pero que no lo son excesivamente, lo que supondría un desperdicio de material y del tiempo necesario para su fresado.

En las operaciones expuestas hasta ahora, la máquina actúa, como ayudante del dibujante, ejecutando órdenes tales como “dibuja un círculo aquí, una línea allí, calcula las tensiones sobre esta parte”. Después “realiza un modelo” del diseño y comprueba si encajan todas las partes, como si se tratase de un modelo de madera. Una vez hecho esto, la máquina dará instrucciones a las máquinas-herramienta dirigidas por computador para que fabriquen las diferentes piezas: “dirígete ahora hacia el radio, fresa esta parte hasta que quede plana, perfora aquí un agujero de 9 mm de diámetro...”.

La máquina también calculará los tamaños y las formas de las piezas de metal huecas necesarias para el fresado. En los sistemas más sofisticados pasará información de las cantidades de materiales que se necesitan al computador que lleva la contabilidad y realiza los pedidos de materiales.

PROCESAMIENTO DE IMÁGENES

Los computadores, aun sin disponer de la capacidad de “ver” realmente, pueden ser de gran utilidad para realzar y manipular fotografías. Actualmente se usan para mejorar la nitidez de imagen en fotografías borrosas, cambiar colores reales por fantásticos y examinar vistas detalladas tomadas desde satélites buscando rocas indicadoras de la presencia de petróleo o bases de misiles. Gran parte de este trabajo se basa en una teoría matemática del barón Jean Baptiste Joseph Fourier (1780-1830).

Para poder apreciar el trabajo de Fourier, necesitamos comprender cómo se logra que un computador “vea”. El problema que se presenta en primer lugar es el de convertir una imagen en bits: 0 y 1. La forma más fácil de resolverlo es tomar con una cámara de televisión una imagen de la escena y después *digitalizada.* Esto implica dividirla en pixeles y medir después la luminosidad en cada uno de ellos. Si se trabaja en color, es necesario medir además la intensidad de cada uno de los tres colores primarios en el pixel; mas, para simplificar las cosas, de momento supondremos que estamos interesados únicamente en imágenes en blanco y negro. Puesto que la cámara de televisión transforma en un voltaje la intensidad luminosa en cada punto, todo lo que se requiere es convertir ese voltaje en un número utilizando un convertidor analógico-a-digital (*analogue-to-digital convertor, o* ADC).

Las cámaras de televisión exploran la imagen en líneas paralelas. Para simplificar, supongamos que el número de filas de pixeles que tendremos es Igual al número de líneas en la imagen de televisión (unas 200 en una cámara de televisión corriente). Si el ADC trabaja a una velocidad tal que realiza 200 conversiones por línea, tendremos 200 × 200 puntos de imagen.

El problema siguiente que hay que plantearse es el de cuántos niveles de luz se desean. El ADC mide una intensidad luminosa y la convierte en bits; la precisión con que la mide depende de la cantidad de bits que produce. La medición más simple posible produce 1 bit; ‘0’ que significa oscuridad o ‘1’ que significa luz. Una conversión de 4 bits da 16 niveles de luz, y una de 8 bits da 256; el diseñador del sistema debe escoger el nivel de precisión que desea, estableciendo un equilibrio entre la velocidad de procesamiento y el espacio necesario para el almacenamiento, por un lado, y el nivel de precisión por el otro.

Cuando introducimos una imagen en la memoria, de hecho hemos conseguido convertirla en algo que en vez de dos dimensiones tiene sólo una.

Es fácil entender las ideas desarrolladas por Fourier si se contemplan en relación con la música, que es asimismo unidimensional. Una nota musical pura consiste en una onda sinusoidal de determinada frecuencia y un diapasón puede producirla con bastante exactitud.

Si se oyen dos diapasones a la vez, se escucharán dos notas, aunque la presión del aire en el tímpano no tiene más que un valor único en cada momento. El oído es capaz de realizar un análisis de Fourier de la onda compleja y separar las dos ondas sinusoidales que la constituyen. Cuando se escucha una orquesta, realiza la misma función para varias docenas de notas puras.

Es fácil comprender que el complicado sonido de una orquesta pueda descomponerse en un conjunto de notas simples; menos fácil de comprender es el hecho de que puede hacerse lo mismo con cualquier sonido o, en realidad, con cualquier forma. Fourier nos enseñó que cualquier función $F(x)$ puede transformarse en una suma de senos y cosenos mediante una ecuación como la que sigue:

$$F(x) = a_0 + a_1\cos(x) + b_1\mathrm{sen}(x) + \ldots + a_n \cos(nx) + b_n\mathrm{sen}(nx) \ldots$$

donde a_n y b_n son los coeficientes de Fourier. Un ejemplo que no resulta en ningún modo evidente es el que ofrece una onda cuadrada: un tren de 1 (unos) binarios. Es difícil ver cómo puede obtenerse, sumando ondas sinusoidales, una onda cuadrada, pero en realidad ello es posible. El análisis de Fourier de la pulsación demuestra que está formada por una onda básica tal que la mitad de su longitud es igual a la amplitud de la pulsación, más una onda de longitud y amplitud 1/3, más una onda de longitud y amplitud 1/5... y así sucesivamente.

Las transformaciones de Fourier constituyen un instrumento de análisis matemático de gran utilidad para realizar operaciones tales como la filtración de frecuencias y la digitalización de la información. No es obligado que las confinemos a operar sobre señales unidimensionales como el sonido; también pueden aplicarse a cosas bidimensionales tales como una fotografía, sobre las cuales, una vez digitalizadas pueden realizarse una serie de operaciones basadas en las transformaciones de Fourier y destinadas a realzar la imagen. Una de las más simples consiste en amplificar las altas frecuencias; en términos de sonido, elevar los tiples. En una imagen, las altas frecuencias se dan en los bordes de los objetos, donde cambian muy rápidamente los niveles de intensidad. Si se amplifican estas altas frecuencias, los bordes se hacen más visibles y la imagen se “encrespa”.

CUADROS MEDIANTE NÚMEROS

Un pintor obtiene sus colores de los tubos de pintura y los distribuye como le parece más conveniente sobre la tela. Un computador los obtiene de una tabla de colores y los distribuye sobre la pantalla como le parece más conveniente al programa. Para los pintores, el computador puede ser una especie de tela dotada de inteligencia.

La mayoría de los cuadros están constituidos por líneas y manchas de color dentro de los contornos definidos por las líneas. En su forma más tosca, el efecto que se obtiene es similar al que producen los “cuadros pintados mediante números”; pero, en definitiva, la mayoría de las obras de arte en la tradición europea tienen este tipo de estructura. El primer paso, o sea, introducir en la máquina los contornos, es el que ofrece mayores dificultades. Por supuesto, el artista tiene el teclado a su disposición para guiar a su “pincel” por la pantalla, pero cabe preguntarse ¿de

qué prestigio gozaría Leonardo si hubiese hecho sus dibujos de la misma manera que los adolescentes cazan Klingons? En plan masoquista se *puede* utilizar el teclado para dibujar; pero el panel de digitalización ofrece una alternativa más interesante: se puede dibujar en él (a menudo con un lápiz electrónico) de forma normal y obtener, además del dibujo sobre el papel, una imagen en la pantalla.

Una vez se ha introducido un contorno en la máquina, puede colorearse guiando el cursor a un área determinada e indicando a la máquina que la coloree de azul magenta con motas de verde y pardo rojizo. Si no se está satisfecho del resultado (lo que es fácil que ocurra), puede cambiarse el color a naranja con motas de gris y violeta.

Sin embargo, el computador puede hacer mucho más que lo expuesto hasta ahora. Con un buen paquete de gráficos se ha de poder constituir una biblioteca de subimágenes tales que, con el tamaño y color elegidos, puedan introducirse en el dibujo final en la posición que se desee. Esto es de gran utilidad para los arquitectos al tener que presentar perspectivas de sus edificios que incluyan elementos tales como coches, farolas, mamás empujando cochecitos de niño, árboles y perros.

Por otro lado, de este modo no se depende exclusivamente de la propia habilidad para el dibujo, ya que el computador, con una cámara de televisión y un digitalizador, puede aceptar imágenes ya existentes. Una vez obtenida la imagen, el artista puede cambiar su textura, perspectiva y colorido. Puede mezclarla con otras imágenes, repetir determinados esquemas y cambiarla hasta el punto de hacerla irreconocible.

Es fácil imaginar que dentro de pocos años, cuando este tipo de sistemas se popularice, las tiendas de arte y de productos gráficos venderán versiones digitalizadas de imágenes estandarizadas. Será posible comprar discos flexibles con la *Mona Lisa o* con imágenes de mujeres en ropa interior y obtener a partir de ellos y de forma automática sorprendentes fantasías visuales en la pantalla del computador personal.

DIBUJOS ANIMADOS

La mejor manera de usar computadores es ponerlos a trabajar en las tareas más estúpidas y aburridas. Si hacen bien su trabajo, se les da después la oportunidad de hacer algo más interesante; pero recuérdese que siempre hay que empezar por lo que resulte más tedioso.

Uno de los trabajos más monótonos del mundo es probablemente dibujar películas de animación. Para un minuto de película se necesitan 1.500 dibujos; para toda una película, varios cientos de miles. Ver trabajar a los dibujantes de dibujos animados es como ver crecer la hierba. No es sorprendente que tan pronto los computadores estuvieron en condiciones de producir dibujos razonablemente buenos, se requiriesen sus servicios para realizar dibujos animados.

En un estudio de dibujos animados los mejores artistas dibujan los momentos culminantes de la acción (Tom mueve la pata hacia atrás; Tom le da una patada a Jerry) y los talentos menores dibujan los numerosos planos intermedios, mostrando el pie de Tom acercándose cada vez más al trasero de Jerry. Este proceso se conoce con el nombre de "llenar huecos" y corresponde al tipo de cosas que las máquinas pueden hacer.

Supongamos que tenemos una vista frontal de un bulldog que se ha encaprichado con un globo (del modo como se encaprichan estos animales). En lugar de dibujar las sucesivas etapas de la "descarga" del bulldog, el sistema de animación llenará los huecos entre los dibujos inicial y final, produciendo las imágenes intermedias tras calcular los puntos proporcionales del recorrido.

Para ahorrar trabajo y evitar inconsistencias, los dibujantes no dibujan cada plano de la película. Los fondos se dibujan una vez y se utilizan después en los sucesivos planos. Los personajes que se mueven frente a ellos están dibujados en hojas de plástico transparente y se sitúan en los lugares correspondientes para después fotografiar el conjunto, obteniéndose así un plano. Un computador es perfectamente capaz de realizar este tipo de trabajo; incluso muchos computadores personales disponen de sistemas para hacerlo, aunque no ofrecen el grado de resolución de imagen que se precisa en las máquinas profesionales.

El computador puede resultar muy útil para colorear: puede dar color a una escena una y otra vez, permitiendo al artista ensayar muchas más combinaciones de las que probaría en el papel.

Gráficos de tres dimensiones

El dibujo de objetos en tres dimensiones, mucho más difícil e interesante que el dibujo plano, es una de las mejores "especialidades" de los

computadores. Estas máquinas son en muchos sentidos asistentes ideales de los delineantes, ya que pueden realizar fácil y rápidamente los laboriosos cálculos necesarios para obtener dibujos en perspectiva.

Consideremos a continuación un dibujo esquemático de una casa vista en perspectiva; es decir, de modo que todas las líneas paralelas horizontales converjan claramente hacia un mismo punto del horizonte. El efecto que ello produce en la forma de la casa es fácilmente calculado y sus elementos pueden dibujarse en la posición correcta. Dado que la casa está formada por planos, todo lo que la máquina tiene que hacer es almacenar las coordenadas de las superficies, lo que también facilita las cosas.

Sin embargo, para obtener una imagen realista en tres dimensiones la casa no puede ser transparente, lo que supone “eliminar las líneas ocultas”. El computador hace esto calculando qué elementos se ven delante de otros y eliminando estos últimos. El proceso es realmente complejo y son muchos los esfuerzos que se han realizado para conseguir programas eficaces para realizar este trabajo.

Una forma tan sencilla como la de las casas no es común en la naturaleza; de hecho, ni siquiera lo es en el mundo construido por el hombre. La mayoría de las cosas poseen formas con curvas más o menos complicadas. El diseñador de software para gráficos en tres dimensiones puede escoger entre dos posibilidades: calcular cada punto de los objetos que deben mostrarse y almacenar las coordenadas de estos puntos, o bien representar pequeñas porciones de superficies y almacenarlas. Dado que almacenar puntos en la memoria resulta terriblemente caro, es mucho mejor (aunque ello exija utilizar gran cantidad de memoria si se quiere hacer un buen trabajo) dividir cada superficie curva en pequeños polígonos planos y almacenarlos. Existe un software especial para suavizar las líneas poligonales y obtener una curva de aspecto natural, pero su ejecución requiere bastante tiempo y no sirve para lograr simulaciones en tiempo real.

Sin embargo, en una imagen la forma no lo es todo. Los objetos reales poseen una textura, unas zonas iluminadas y otras en sombra, un color, y muestran el reflejo de otros objetos. El computador puede calcular todas estas características y “pintar” sobre la imagen los correspondientes efectos.

Por ejemplo, la textura de un ladrillo es bastante diferente de la del vidrio. Para producir un efecto perfecto, el programa de gráficos en tres

dimensiones debe calcular de dónde viene la luz, cómo se reflejará en la superficie y adónde irá a parar. Además, la imagen puede no ser estática, en cuyo caso el computador debe ser capaz de calcular el movimiento de las cosas que figuran en ella (las olas se mueven de acuerdo con leyes físicas y el programa debe asignar al agua que forma las olas un "peso" tal que éstas se muevan del modo adecuado).

No es sorprendente que imágenes de este nivel de complicación exijan varias horas de procesamiento.

Como consecuencia de lo expuesto, los computadores se emplean cada día más como asistentes eficaces en el proceso de creación artística. En primer lugar pueden, como podría un asistente torpe, cambiar un color verde por un color azul o trazar un dibujo a partir de un croquis preliminar. En la actualidad pueden realizar una gran parte de las tareas que está obligado a realizar el artista, de tal modo que éste puede limitarse a especificar en términos generales un paisaje (aquí una montaña, aquí un valle, allí una llanura ondulada y, finalmente, un lago), mientras se deja que la máquina cree los contornos del territorio (teniendo en cuenta las leyes de la geología), lo vista con la vegetación adecuada, la coloree de acuerdo con la estación del año de que se trate y lo sitúe todo bajo la luz adecuada a la hora y circunstancias climatológicas del día y del año. ¿Cuál es, de acuerdo con todo esto, la tarea que debe realizar el artista?; concebir la idea original, que es probablemente lo que tiene mayor mérito.

LA VISIÓN DE LOS COMPUTADORES

En las páginas 168-170 hemos visto que es posible reducir una imagen a pixeles y almacenarla en la memoria de un computador. Las dificultades que presenta conseguir sistemas para que la máquina "vea", es decir, que sea capaz de interpretar la imagen que se le ha introducido, evidencian que, después de todo, el cerebro humano es realmente mucho más inteligente que el mejor de los computadores.

El primer problema con el que se enfrenta un analista de imagen es el del ruido eléctrico. Incluso si se mostrase a la cámara una hoja de papel de color gris homogéneo, que llenaría la memoria con pixeles del mismo valor, por ejemplo 127, se producirían desequilibrios. El valor medio puede ser 127, pero el que corresponde a cada célula en particular fluc-

tuaría en torno a ese valor. A algunas les correspondería 126, a otras 128. Incluso algunas podrían corresponder a valores tan alejados del medio como 130 o 124. Estas variaciones son debidas al llamado principio de incertidumbre, que rige el comportamiento físico a escala atómica de la cámara, los amplificadores de imagen y el convertidor analógico-a-digital. Para combatir el ruido, el sistema debe recorrer primero la imagen, comparando cada célula con las ocho células vecinas más próximas. Si el valor que corresponde a una célula difiere mucho del medio, se reemplaza por este último.

El ojo y el cerebro humano analizan las imágenes por diversos procedimientos. Extraen los contornos; encuentran áreas de tono similar; utilizan las sombras y muchas leyes básicas de la física (o del sentido común). También tenemos otra ventaja respecto al computador y es que podemos comparar las visiones de los objetos que nos llegan a través de cada uno de los dos ojos y que difieren ligeramente entre sí, es decir, que disponemos de visión estereoscópica; pero esto sólo es útil si los ojos, cada uno por su lado, han interpretado las masas que tienen frente a ellos.

Lo primero que hace un computador para interpretar lo que “ve” es un análisis de los contornos. El objeto de este análisis es convertir la imagen en un dibujo lineal. Se realiza repasando de nuevo los pixeles para identificar las células que difieren significativamente del promedio o de sus vecinas. Estas células se guardan y las otras se eliminan. Como resultado del proceso se obtiene (o se obtendría, si el mundo fuera perfecto) un dibujo del contorno de los objetos presentes en la escena. Lo que se pretende conseguir son lazos cerrados: cualquier objeto sólido presente en la escena debería tener un contorno que empiece en un punto, lo recorra y termine en el mismo punto.

Por desgracia, en la práctica el contorno se rompe y se “despista”. Se rompe cuando las intensidades a ambos lados de una línea son iguales, con lo que el proceso de “promediación” no puede hallar diferencias; se despista cuando, a causa de reflejos, se producen diferencias de intensidad no significativas. Las sombras representan un terrible problema porque es imposible analizarlas si no se sabe dónde se encuentra el Sol. A estas dificultades se suma la debida al hecho de que los contornos de los objetos más próximos rompen los contornos de los más lejanos.

Supongamos que, a pesar de todo, llegamos a disponer de un buen croquis. Seguiremos sin tener el contorno completo de todos los objetos

presentes en la escena, ya que el lado más lejano de cada objeto está oculto por su lado más próximo y los objetos que están más cerca ocultan a los que están más lejos. Respecto al primer problema es bien poco lo que puede hacerse, ya que no podemos ver el lado más lejano de ningún objeto. Para el segundo, puede aplicarse una técnica conocida como "análisis de vértices".

El análisis de vértices aprovecha el hecho de que muchas de las cosas que normalmente se miran con un sistema de computador poseen bordes y esquinas rectos. Los vértices se ajustan a las leyes de la geometría y la máquina, al examinar los bordes y las esquinas, puede separar las esquinas verdaderas de las que corresponden a accidentes producidos por la superposición de dos objetos diferentes.

Utilizando el análisis de esquinas y vértices, la máquina puede progresar en el sentido de llenar los huecos de sus contornos y de distinguir los contornos de un objeto de los de otro.

Una segunda técnica para interpretar imágenes es encontrar áreas que tengan el mismo tono. Un método es elegir un pixel al azar y comprobar si las células a su alrededor tienen valores significativamente próximos. Si ocurre así, el procesador las marca. Este proceso, reiterado, define áreas que tienen intensidades similares y que probablemente forman parte de los mismos objetos. Cuando el programa no puede encontrar nuevas células que se correspondan, escoge otro punto de partida e intenta definir otra área. Un programa que utilizase las dos técnicas expuestas buscaría la correspondencia entre áreas del mismo tono con contornos, usando la información derivada de los tonos para llenar los huecos de los contornos.

Otra manera de seleccionar lo que se ve se basa en la aplicación de leyes físicas de carácter muy sencillo. Sabemos que las cosas deben tener soportes, no pueden ser demasiado pesadas o voluminosas en su parte superior. Tenemos una idea acerca del grosor de las paredes, árboles y ramas. Las cosas muy bajas y planas o muy largas y delgadas son extrañas; suponemos que son consecuencia de una ilusión óptica, un análisis erróneo de la escena. Sabemos que cuando las cosas se mueven tienden a hacerlo en línea recta, que si dan la vuelta a esquinas tienen que hacerlo siguiendo curvas suaves. Si, en un camino, a bastante distancia delante de nosotros, vemos una mancha brillante situada verticalmente sobre una mancha oscura, podemos suponer que se trata de una mujer que lleva un

sombrero de color claro y un vestido oscuro. Si, en este caso, las dos manchas se separasen o si empezaran a moverse a gran velocidad, nos llevaríamos una gran sorpresa.

También disponemos de una amplia biblioteca de imágenes almacenadas. Sabemos perfectamente qué aspecto tienen las personas y cómo están constituidas. Conocemos gran número de objetos de todas clases (mesas, sillas, coches, botellas, refinerías de petróleo, barcos, flores, insectos) y, para interpretar lo que vemos, superponemos constantemente imágenes que tenemos almacenadas. Cuando gritamos "Es un pájaro. Es un avión. No, ¡es Supermán!" estamos aplicando esas imágenes almacenadas a una mancha que se mueve en el cielo para tratar de identificarla. Cuando la mancha se aproxima y disponemos de más información, tenemos que corregir las apreciaciones.

Por desgracia, el computador está muy lejos de poder realizar un proceso semejante. Cada vez que mira el mundo tiene que empezar de nuevo como si nunca antes hubiese visto nada. Un área de aplicación de los computadores con visión potencialmente muy amplia es la industria, el trabajo en fábricas, minas o en el interior de reactores nucleares.

Una tarea aparentemente muy simple pero de gran utilidad consiste en identificar las piezas que se necesitan para una máquina, de manera que se pueda coger una pieza y ensamblarla en el momento justo. Pero incluso esto es difícil. El primer problema es que el computador sólo puede trabajar con imágenes bidimensionales. Para evitar problemas derivados de la acumulación de polvo, de las sombras y de la presencia de metales de distinto color, las imágenes normalmente sólo presentan los contornos; las piezas se sitúan frente a un fondo iluminado y se muestran a una cámara de televisión. Muchas piezas de máquinas, tales como los engranajes, son esencialmente planas; pero muchas otras tienen formas sólidas que podrían situarse frente a la luz de muchas maneras diferentes y ofrecerían al computador varios contornos distintos. Una posibilidad de solución de este problema es tratar cada orientación de la pieza como si fuese una pieza distinta que hay que coger y hacer girar de modo distinto para ensamblarla de forma adecuada al conjunto

Cuando al computador se le presenta un contorno, lo intenta reconocer de varias maneras. Primero puede medir el tamaño del objeto. Después puede calcular determinado número de índices acerca de la forma del objeto. En un programa reciente de la *Machine-Intelligence Unit* de la

Universidad de Edimburgo, la máquina reconocía chocolatinas (escogidas por sus formas que, aunque parecidas, presentan sutiles diferencias) midiendo cosas tales como el área de la forma, su perímetro, su anchura máxima y mínima, etc. Se le presentaron chocolatinas de caramelo, cereza, turrón, mantequilla, naranja y coñac, y un sistema experto elaboró una regla para identificar los diferentes tipos a partir de diversas permutaciones de las medidas.

COMPUTADORES QUE HABLAN

En el mundo de los computadores hay quien sostiene que estas máquinas no serán realmente útiles para nadie hasta que sean capaces de hablar y comprender el lenguaje hablado. La segunda de estas condiciones es, según veremos, difícil de satisfacer; la primera no presenta dificultades irresolubles. Los computadores que hablan tienen utilidad en situaciones en que sus usuarios no pueden leer, sea porque son ciegos o demasiado jóvenes, o no desean tener que hacerlo porque están ocupados haciendo otra cosa: pilotando un avión, conduciendo un coche, caminando por un almacén examinando los stocks.

En todos estos casos sería útil que el computador pudiese hablar; quizá leyendo en voz alta el libro de instrucciones incorporado a su memoria, en los dos primeros casos, y dando advertencias, comentarios o confirmaciones en el tercero.

Existen dos maneras distintas de efectuar la grabación digital y la reproducción del lenguaje hablado. La primera consiste simplemente en tomar los niveles de voltaje variables producidos por un micrófono, extraer muestras de los mismos a unos 8 kHz, digitalizar los resultados y almacenarlos como una serie de bytes. Esto funciona perfectamente, pero exige unos 8 kilobytes de memoria para cada segundo de lenguaje hablado. Un disco de 5 MB almacenaría sólo diez minutos.

Este sistema no se puede poner en práctica en la realidad. Para comprender cómo podemos obtener un sistema mejor, tenemos que considerar primero la mecánica de la boca. Después de muchos años de investigación se dispone de un modelo conceptual fiable de los mecanismos productores de sonidos básicos de la boca y la garganta. El sistema vocal humano consiste en una especie de tubo variable, la boca, que se estrecha y remata hacia abajo en la garganta. En la garganta y la lengua hay

músculos que pueden alterar el tamaño y, por tanto, la frecuencia de resonancia en este espacio.

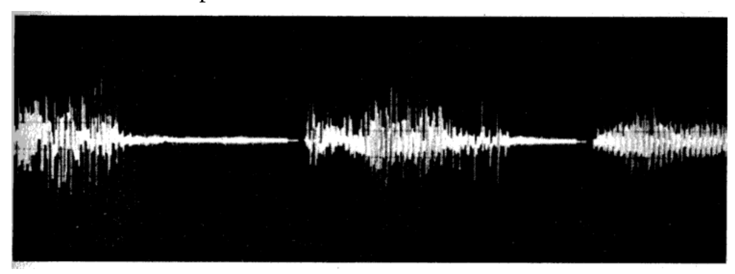


Fig. 24. La compleja forma de las ondas del habla. Algunas palabras dichas frente a un micrófono, produjeron esta traza en una pantalla de rayos catódicos. Los sistemas de computadores que intentan comprender el lenguaje hablado tienen que analizar estas trazas en varias bandas de frecuencia y después buscar en su memoria, entre las palabras que «conoce», aquellas que posean parámetros que se correspondan con los de las palabras dichas frente al micrófono.

Los sonidos de las consonantes consisten principalmente en chasquidos, silbidos y detenciones impuestas a las vibraciones del oscilador o columna de aire. Para imitarlos todo cuanto se necesita es una fuente de ruido "blanco", es decir, un ruido formado por cantidades iguales de todas las frecuencias audibles. Puede obtenerse fácilmente por medios electrónicos amplificando la salida de un diodo Zener.

Existen dos dispositivos para producir sonidos: las cuerdas vocales (que son en realidad plegamientos de la piel) en la laringe, que pueden estirarse más o menos de manera que producen zumbidos de distinta frecuencia cuando el aire procedente de los pulmones pasa a través de ellas; y la lengua y los dientes, donde se produce un silbido cuando el aire pasa a través de ellos. Estos dos componentes pueden imitarse fácilmente con un sistema electrónico. Las cuerdas vocales zumban a frecuencias que vienen determinadas por el tamaño y la forma variables del resto de la boca.

Esta parte del sistema puede imitarse con un generador de frecuencia variable, que haga el papel de las cuerdas vocales, y un conjunto de tres filtros, controlados digitalmente, que produzcan efectos similares a los de las diferentes cavidades de resonancia que pueden formarse en la boca. Así pueden obtenerse todos los sonidos de las vocales (aunque palabras

tales como "día" o "soy" exigen un rápido cambio en la frecuencia básica) producidas en la boca moviendo la lengua. Esto se consigue electrónicamente alterando la frecuencia a media vocal.

También existen algunos sonidos nasales (m, n, ñ) que son producidos por la resonancia característica de la nariz a unos 1.400 Hz. En un computador exigen un filtro independiente. Y, finalmente, las fricativas (b, d, g, f, o, s, l, y, x, z) que son producidas por la lengua y los dientes sin que exista resonancia en el área vocal. Para transformar el lenguaje escrito en lenguaje hablado, se necesita una gran habilidad, ya que han de producirse los chasquidos, vibraciones, silbidos y psss particulares necesarios para decir, por ejemplo, «¿Cómo está usted?» en términos de las órdenes que daría el sistema nervioso a los músculos de la garganta de una persona que habla; además, ha de hacerse en el tiempo que tardaría esa persona en articular esos sonidos, lo que resulta mucho más difícil que simplemente almacenarlos. Una vez se ha conseguido la transformación anterior, el ahorro de datos es espectacular. La gente habla normalmente a razón de 100 palabras por minuto y las palabras tienen como término medio en inglés 6 letras, de manera que para almacenar de este modo lenguaje hablado se necesitan unos 10 bytes por segundo (una seiscientava parte de los datos necesarios para el lenguaje digitalizado directamente).

Existen diversos chips para este tipo de trabajo: uno de ellos, el SC-01 ofrece consonantes inglesas (f, t, g, k, z, b, d, j, p, h, m, n, 1, r, w, th, sh, ch) y vocales inglesas (a, ah, ae, aw, e, eh, i, o, oo, u, uh, y) todas ellas en dos o más longitudes, que se indican mediante subíndices. Por ejemplo, existen cuatro sonidos eh, cuyas longitudes van desde los 59 a las 185 milésimas de segundo, y tres pausas también de diferente longitud. Si se suministra un texto inglés normal al SC-01, lo que se obtiene es en general desastroso. "Escribir" para este chip no es sencillo. Por ejemplo, la palabra "Computer" deberá escribirse así: "K UH_3 M P U_1 T ER", "Good afternoon" (buenas tardes) será "G OO D AH_1 F T UH_3 NU U_1 N".

Otro problema es el que presentan los acentos locales. Lo que suena perfectamente normal para un tejano puede a menudo parecer una jerga incomprensible para un escocés. (El juguete de Texas Instruments "Speak and Spell", que se basa en una tecnología ligeramente distinta de

la expuesta, no tuvo mucho éxito en Inglaterra porque nadie era capaz de entender el acento americano.)

El último problema que todavía no ha sido satisfactoriamente resuelto, es que en una conversación gran parte del significado de lo que se dice no reside tanto en las palabras como en la entonación y el énfasis con que se dicen. De hecho, con frecuencia no necesitamos oír las palabras, nos basta con "ver" cómo las dicen las personas que hablan. Para tratar esto en la forma adecuada se necesitaría que los computadores fuesen capaces de "entender" a partir de la tendencia general de la conversación si se trata de un diálogo agresivo, tranquilizador, afectuoso, aburrido, etc., y de incorporar a sus palabras el tono que mejor se ajuste a las circunstancias. De momento nadie tiene idea de cómo lograrlo.

COMPUTADORES DIRIGIDOS POR LA VOZ

Ya hemos visto que se puede lograr que los computadores hablen, aunque muy imperfectamente. El problema, como vimos, es realmente difícil; pero lograr que los computadores sean capaces de escuchar y comprender la voz humana lo es aún muchísimo más. De hecho, nadie sabe hasta qué punto es realmente difícil, por la sencilla razón de que nadie lo ha resuelto todavía.

Para nosotros es tan natural comprender el lenguaje hablado que subvaloramos el nivel de dificultad que esto puede suponer para una máquina. Las dificultades se presentan a diversos niveles que, además, se refuerzan mutuamente entre sí. El primer problema es transformar los confusos y poco claros sonidos que producimos con nuestras bocas en la relativamente clara representación escrita. Si se intenta realizar una transcripción de una conversación grabada en una cassette es fácil darse cuenta de las dificultades que encierra comprender qué dicen exactamente las personas que hablan; sin embargo, quienes se encontraban presentes en el momento de la conversación probablemente no tuvieron ningún problema en este sentido. El lenguaje hablado está lleno de "aos" y "las"; palabras que en el lenguaje escrito estarían separadas por un espacio, se articulan sin discontinuidad; otras tienen una discontinuidad en donde de hecho no existe. El problema no tiene nada de sencillo. Para decidir qué sonidos constituyen una palabra se necesita un gran conocimiento del tema de la conversación. Un programa de computador destinado a trans-

cribir lenguaje hablado en escrito debería ser capaz de "comprender" lo que se está hablando.

Comprender, en este sentido, es uno de los grandes problemas de nuestra época. Es evidente que no basta con disponer simplemente de un diccionario. Cuando una persona nos dice "¿todavía usas este trasto para lo mismo que antes?" es imposible que entendamos lo que nos quiere decir si no hemos prestado atención a lo que nos ha dicho anteriormente. Y esto suponiendo que todo lo que se necesita para comprender la conversación nos ha sido explicado previamente, que podemos seguir los pensamientos que se nos comunican simplemente escuchando. Pero, por supuesto, raras veces es así. A menudo la gente señala hacia algo y dice "eso de allí no, ése no, el otro", lo que puede dejar helado al transcriptor.

Pero, además, no sólo se utilizan simplificaciones de lenguaje cuando se hace referencia a cosas que otros pueden ver; con frecuencia se supone el conocimiento por parte del que escucha acerca de todo tipo de cosas ocurridas en el pasado. Parte de este conocimiento es tan general que entra dentro del campo de conocimientos que todo el mundo posee; por ejemplo, que un amor no se encuentra en cada esquina y que cuando se habla de un lugar para fijar una cita es mucho más probable que la palabra utilizada sea "allí". Pero en muchos diálogos gran parte del significado puede basarse en experiencias compartidas por las personas que hablan y no ser accesible a otras.

Incluso si se hubiese logrado resolver el problema de proporcionar al computador suficientes conocimientos para comprender conversaciones, todavía no sabríamos cómo conseguir que la máquina halle el significado correcto, de entre todos los posibles, de un nuevo sonido. Lo que necesitamos es una sofisticada estructura de proceso que ofrezca posibles interpretaciones de sonidos a otro procesador que está construyendo una imagen de la conversación; éste aceptará o rechazará las interpretaciones que se le ofrecen según se ajusten o no al esquema de que ya dispone en función de lo que se le ha comunicado hasta el momento.

El mejor sistema hasta el momento es un prototipo de los laboratorios Bell, que habla con la gente por teléfono acerca de reservas para vuelos aéreos. No precisa saber mucho acerca del mundo, aparte de los horarios de vuelos y la estructura de las frases sencillas. Cuando una voz al otro lado de la línea le dice: *"I want to fly to San Francisco on Saturday"* ("Quiero volar a San Francisco el sábado"), no necesita buscar mucho

para averiguar de qué se trata. Divide en palabras lo que ha oído, escuchando las pausas. También divide las palabras en “marcos” de 15 milésimas de segundo de longitud. Dentro de un marco la frecuencia y el volumen de sonido permanecen prácticamente invariables, de manera que la máquina puede describir las palabras en función de estas muestras.

Divide los sonidos en bandas de frecuencia de una octava (el doble de la frecuencia) de ancho cada una (100-200, 200-400, 400-800, 800-1.600, 1.600-3.200 Hz) y anota la intensidad de sonido en cada banda de cada marco. Una vez ha reducido de este modo cada palabra a un conjunto de números, puede intentar encontrar en su memoria las palabras que se corresponden con las oídas. Quizá no encuentre ninguna palabra en la memoria a la que correspondan exactamente los números que ha hallado, pero como en una conversación acerca de horarios de aviones son muy pocas las palabras que se utilizan, la máquina puede establecer la correspondencia entre la palabra oída y un número muy limitado de candidatas. Una vez ha obtenido estas palabras candidatas, está en condiciones de encajarlas en la estructura de la frase. En la frase anterior, quizás “I” (yo) y “to” (a) fueron claras, pero la palabra entre estas dos lo fue menos. El conocimiento que de la sintaxis inglesa tiene el sistema le permite especular que las frases comunes probables que empiezan con una “w” en esa posición son “would like to” (me gustaría) o “want to” (quiero), y como no hay suficiente espacio para que quepa la primera, la palabra debe ser “want”.

La transferencia de esta tecnología a temas más amplios presenta nuevos problemas aparte del enorme incremento en el número de posibles significados para cada palabra confusa. Para simplificar la investigación de palabras, el sistema supone la existencia de una gramática (un conjunto de reglas) que la persona que habla debe seguir para enlazar sus palabras tales como: “María (sujeto) tenía (verbo) un corderito (complemento directo)”. Las mismas estructuras rigen para “Yo (sujeto) le di (verbo) un beso (complemento directo)”.

Sin embargo, los estudios sobre gramática realizados a lo largo de muchos años han demostrado que la apariencia de simplicidad de este tipo de reglas no corresponde a la realidad.

A medida que se profundiza, resulta cada vez más claro que cada palabra particular tiene su propio conjunto de reglas gramaticales, lo que complica enormemente la investigación de palabras para el sistema. Ya

no puede decirse “la próxima palabra ha de ser un verbo, por tanto me limitaré a buscar en mi diccionario palabras que sean verbos” porque pueden existir uno o dos nombres propios que encajen en la frase en esta posición. La lingüística informatizada ha provocado el desánimo en muchos investigadores.

SENSORES

Para que un computador pueda actuar de forma útil sobre el mundo material, es necesario que disponga de instrumentos que le proporcionen información acerca de ese mundo; tales instrumentos reciben el nombre de sensores. La mayoría de ellos se basan en algún tipo de dispositivo eléctrico que transforma el efecto físico de modo que pueda ser medido en un voltaje, que, a su vez, un convertidor analógico-a-digital transforma en formato binario.

La mayor parte de los sensores que se utilizan en la industria y en los proyectos científicos y militares, están destinados a determinar la posición, la distancia, la velocidad, la aceleración y la fuerza.

Posición

Con frecuencia es necesario determinar la posición de algún objeto o pieza de una máquina. Un caso muy sencillo es el de los sistemas de alarma antirrobo que deben determinar si una puerta o una ventana están abiertas o cerradas. Esto puede lograrse instalando un microinterruptor en contacto con la puerta o la ventana. Sin embargo, estos interruptores se deterioran fácilmente y no son fiables, por lo que normalmente se utilizan dispositivos sin contactos. Por ejemplo, una célula fotoeléctrica, en que el haz de luz es interrumpido por una pequeña paleta al cerrar la puerta. Sin embargo, las lentes pueden ensuciarse, por lo que resulta todavía más interesante emplear un sensor magnético. Este tipo de sensores se basan en el denominado efecto Hall, que se apoya en el hecho de que un campo magnético altera las propiedades de los transistores. De acuerdo con esto, para determinar si una ventana está cerrada, se instala un imán en la ventana y un detector en el marco. Un pequeño circuito

electrónico detecta cuándo está cerrada la ventana y lo indica así al computador.

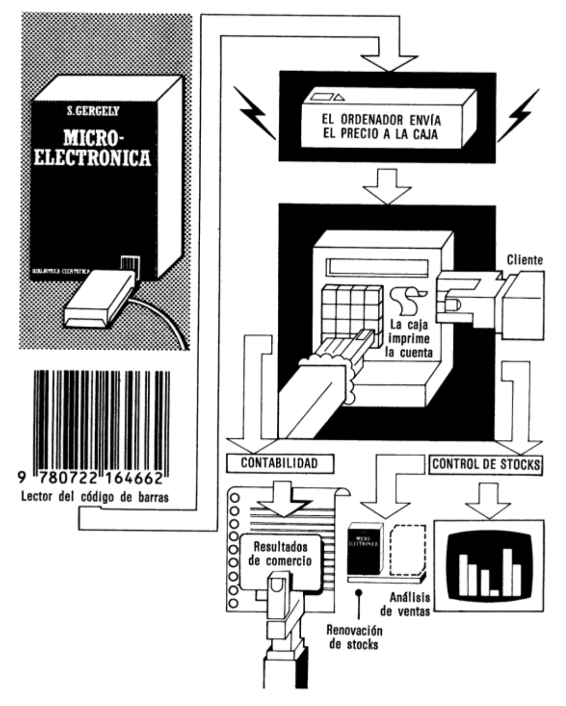


Fig. 25. La informatización de una librería se inicia con la incorporación de códigos de barras a los productos. Un sensor lee estos códigos mediante un haz luminoso y esta información es transmitida a un computador que envía a caja el precio y la descripción del artículo para que ésta imprima para el cliente la cuenta. El cliente paga en la forma usual, pero el computador utiliza asimismo esta información para actualizar los registros de stocks y realizar los cálculos de contabilidad. Puede renovar pedidos mucho más rápidamente de lo que es posible hacerlo con un sistema manual. Al mantener un control de los stocks artículo por artículo, el sistema pone las cosas más difíciles a los rateros. El paso siguiente en el desarrollo de la informatización de los comercios consiste en que la caja pueda leer la tarjeta de crédito del cliente y el computador envíe un mensaje a su banco para que se cargue a su cuenta corriente el valor de las compras realizadas.
En muchos países del mundo, las tiendas y los bancos trabajan ya conjuntamente para eliminar el dinero en efectivo de las transacciones cotidianas.

En muchos casos lo que se desea saber es qué distancia ha recorrido una determinada pieza de una máquina en su trayectoria. La solución más simple consiste en unir la pieza móvil al cursor de una resistencia variable y medir el valor de esa resistencia.

Sin embargo, una resistencia puede alterarse, por lo que es más práctico utilizar un sistema óptico que incluye una especie de reja, que pasa frente a un foco luminoso, y un detector situado al otro lado: cuando la pieza en su movimiento se desplaza una distancia igual a la anchura de una línea en la reja, el sistema envía una pulsación al computador. El mismo tipo de dispositivo puede emplearse para medir rotaciones.

Sensores a distancia

A menudo es necesario medir distancias y grosores sin que nada pueda tocar el objeto medido. Una solución posible es enviar una pulsación de ondas ultrasónicas para que reboten en la superficie del acero y medir el tiempo de retorno: cuanto más grueso sea el acero menor será este tiempo. Otro método sería enviar un haz de rayos X a través del acero y medir la intensidad de la radiación al otro lado; cuanto más grueso sea el acero, menos intensos serán los rayos X que pasan al otro lado.

Los barcos y los yates utilizan desde hace muchos años ultrasonidos para medir la profundidad del agua, simplemente midiendo el tiempo que tarda en regresar una pulsación reflejada desde el fondo del mar. El mismo sistema puede usarse para determinar los volúmenes contenidos en grandes tanques de productos químicos, petróleo o aceite. Los geólogos utilizan explosiones para investigar la estructura de las rocas subterráneas, con el fin de detectar la posible existencia de petróleo.

El principio de la medición del plazo de tiempo que tardan en llegar las ondas de radio se utiliza en la navegación. En los sistemas de navegación modernos, tanto civiles como militares, un barco o un avión pueden calcular su posición a partir de las diferencias entre los tiempos que tardan en llegar señales procedentes de distintos satélites.

Aunque los computadores no pueden propiamente “ver”, son de gran utilidad para mejorar la visión humana. Actualmente, los satélites cartográficos observan la superficie terrestre con cámaras ajustadas para diferentes bandas de frecuencias. Las imágenes resultantes pueden ser anali-

zadas y combinadas por un computador para obtener mucha más información de la que cualquiera de ellas por sí sola podría proporcionar. También puede utilizarse un computador para extraer de una fotografía mucha más información de la que a primera vista parece contener.

Velocidad y aceleración

No es muy difícil medir la velocidad. En un vehículo sobre ruedas, puede usarse un sensor en posición angular para contar barras por segundo y a partir de aquí medir la velocidad de rotación de la rueda y la velocidad del vehículo. Si se trata de un avión, puede medirse la presión de la corriente de aire, o bien puede enviarse una señal de radar para que rebote en el suelo y medir en la señal de retorno el efecto Doppler provocado por el movimiento de avance del aparato. Para medir la velocidad con que fluye un líquido en un tubo, a menudo se emplea una pequeña hélice con un imán en una de sus palas que actúa sobre un sensor de Hall en cada rotación.

No hay ninguna manera de medir una velocidad sin referirse al mundo exterior. Sin embargo, la aceleración puede medirse mediante instrumentos contenidos totalmente en el interior del vehículo, y a partir de la aceleración es fácil calcular la velocidad en un tiempo determinado. Existen dos tipos de acelerómetros: los que miden las aceleraciones lineales y los que miden las angulares.

Medir la aceleración lineal es sencillo: todo lo que se necesita es una balanza. Imagínese una balanza con un peso de 4 kg en su plato, colocada en el suelo de un ascensor. Al empezar a subir el ascensor, la balanza indicará durante breve espacio de tiempo un peso de, por ejemplo, 5 kg. Cuando el ascensor, al llegar al final de su trayecto, disminuye de velocidad, la balanza indicará un peso de 3 kg. Es posible conectar a la balanza un computador que calcule a partir de estas mediciones las aceleraciones y las velocidades del ascensor en cualquier momento y en qué posición se encontraba cuando tenía una aceleración y velocidad determinadas.

Las mediciones de la aceleración angular se efectúan mediante giroscopios. Como sabe cualquiera que haya jugado con una peonza, estos objetos se mantienen en equilibrio con su eje en posición vertical y oponen resistencia a cualquier intento de cambiarlos de posición. Midiendo

estas fuerzas de resistencia puede determinarse la aceleración angular. En los giroscopios más sofisticados, se utilizan rayos de luz. Se envía una pulsación de láser para que recorra una trayectoria circular cerrada entre tres espejos y se detecta después de haber dado varias vueltas. Si el dispositivo (o el vehículo en que se encuentra) ha girado cuando la luz efectuaba su recorrido, la pulsación llegará al detector adelantada o retrasada, con lo que puede calcularse la velocidad de giro. Una vez conocida ésta, puede calcularse cuánto se ha girado; si se sabe en qué dirección se viajaba al principio del vuelo, será posible conocer en qué dirección se viaja en cada momento. Los acelerómetros lineales y angulares de estos tipos se utilizan para determinar la velocidad y posición de los misiles balísticos intercontinentales. Pueden ser tan exactos que, después de un vuelo de miles de kilómetros, el proyectil aterrice a pocos metros de su objetivo.

Fuerza

La forma más fácil de medir una fuerza es hacer que comprima un muelle y después medir la longitud del muelle. Imagínese que se desea medir la fuerza de empuje de un motor a propulsión. El muelle más sencillo que puede elegirse es un montante que aguante el empuje que el motor proporciona al aparato. Puede comprimirse, al igual que un muelle, pero, por supuesto, sólo a nivel microscópico. Para determinar electrónicamente hasta qué punto se comprime, se pega al montante una pequeña lámina de metal que se utiliza como resistencia eléctrica. Al comprimirse el montante, su longitud se reduce, con lo que disminuye la resistencia eléctrica de la lámina de metal. El computador puede medirla y determinar a partir de ella la fuerza de empuje del motor.

Gran parte del misterio del mar tiene su origen en la dificultad de los cálculos necesarios para dar la vuelta al mundo en barco. Con la llegada de los computadores el misterio está empezando a desaparecer.

Son varios los cálculos que un computador puede hacer en un yate, mas para hacerlos tiene que recibir las informaciones pertinentes. Las cosas que evidentemente hay que medir son: la velocidad de crucero; el rumbo del barco dado por la brújula; la velocidad relativa del viento y la dirección. Una vez se han recogido estos datos, pueden utilizarse para

realizar diversos cálculos de interés. A los navegantes lo que más les interesa es saber su posición. Pueden conocerla orientándose a partir de los faros y de los accidentes geográficos de la costa, pero quizá no haya ninguno en el horizonte o quizá sea de noche o haya niebla, por lo que necesitan mantener siempre una "estimación de rumbo", es decir, un gráfico del rumbo y la velocidad desde el punto en que empezaron la travesía hasta el punto en que deberían encontrarse en el momento actual. Un computador al que se haya informado del rumbo dado por la brújula y de la velocidad de crucero puede trazar este gráfico sin dificultad. Una instalación para la estimación del rumbo sería de gran utilidad si se produjese la desgracia de que alguien cayese por la borda; el timonel no tendría más que apretar un botón de nueva puesta en marcha de la estimación de rumbo en el momento en que se da la alarma y volver hacia atrás hasta la posición en que la estimación de rumbo sea nula.

Lo que les interesa conocer a continuación a los navegantes es la verdadera velocidad y dirección del viento, lo que tampoco es difícil de calcular, sabiendo la velocidad de crucero, el rumbo indicado por la brújula y la velocidad y dirección aparentes del viento.

La ley de Murphy aplicada a los barcos de vela afirma que se emplean tres cuartas partes del tiempo navegando lenta y dificultosamente a barlovento. Depende de la pericia del timonel que se avance o no a barlovento con la máxima velocidad posible; y esto no es nada fácil de comprobar, excepto descubriendo después de varias horas de pasar frío que no se han realizado los progresos que se esperaban. El computador puede calcular la velocidad de la embarcación contra el viento y visualizar el resultado como porcentaje de la que sería posible avanzar.

Los entusiastas de las carreras pueden utilizar este dispositivo como cronómetro que les indique cuándo se realizará el disparo que señala el comienzo de la prueba y pueden registrar en cinta las cuatro variables mencionadas, juntamente con los movimientos del timón para su posterior análisis.

Sin embargo, con esto no se agotan los posibles usos del computador. Lejos de la costa, los navegantes han de hallar su posición midiendo la altura del Sol y de las estrellas sobre el horizonte. Una vez medidos estos ángulos con un sextante, los navegantes deben realizar cálculos bastante complicados (utilizando voluminosos libros de tablas) para descubrir

dónde se encuentran. El computador puede realizar estos cálculos con gran facilidad.

SERVOS

La medición de variables del mundo exterior es importante para que los computadores puedan actuar como máquinas que controlan otras máquinas, pero el papel crucial en este aspecto corresponde sin embargo a los “servo bucles”. Para comprender lo que son, daremos un ejemplo sencillo: el de una persona que conduce un automóvil por una carretera recta.

Son muchos los factores que pueden impedir que se conduzca en línea recta: la dirección del coche puede ser defectuosa; pueden existir baches en la carretera que desvíen el coche de su trayectoria; quizá pasen camiones que “absorban” el coche hacia un lado u otro; el conductor puede mirar hacia determinado punto a un lado de la carretera y sus manos sobre el volante seguir, sin que él se dé cuenta, la dirección en que miran sus ojos; pueden producirse golpes de viento que empujen el coche a un lado de la carretera.

Por otra parte, es evidente que no puede escribirse un programa para la conducción del coche sin un conocimiento preciso de las actuaciones de los demás usuarios de la carretera. Si un camión pasa una fracción de segundo antes o después, el programa puede causar un accidente mortal. Lo que se necesita es algo mucho más flexible, que pueda adaptarse a condiciones variables e imprevisibles. Veamos lo que este “algo” tiene que hacer.

El objetivo del conductor es conducir su vehículo en línea recta por el centro de su carril. La única fuente de información que consideraremos es la que viene dada por la visión del conductor, que puede, de forma aproximada, apreciar hasta qué punto el coche se encuentra desviado respecto a la línea que hay que seguir.

El computador debe tratar el problema igual que lo hace una persona: realizando sobre la marcha pequeños experimentos. Cuando se conduce por primera vez un coche, se prueba la dirección y los frenos para ver cómo funcionan. Lo mismo debe hacer la máquina: si cambia el viento, debe darse cuenta y adaptarse. La técnica que permite hacer esto es el servo bucle.

El servo bucle para el control de un coche es muy sencillo. El conductor observa la carretera y calcula adonde desea ir a partir de factores tales como la posición del borde de la carretera, de la línea del centro, y de otros vehículos. Después estima la *diferencia* entre donde se encuentra y a donde desea ir y gira las ruedas de manera que esta diferencia se reduzca a cero. En otras palabras, si el coche se ha desviado un poco hacia la izquierda, gira las ruedas un poco hacia la derecha. Si se ha desviado mucho hacia la izquierda (o si la carretera tiene una curva hacia la derecha), gira las ruedas mucho hacia la derecha. A continuación se da un programa en BASIC de Microsoft para simular este problema[10].

```
5 K1=0,05:K2=–2:K3=0,8:K4=0,2
10:INPUT"VIENTO, KPH";W
20 PRINT "RUEDAS, ANGULO, GRADOS";
40 PRINT TAB (25);"DESVIACION DEL COCHE, METROS";
50 FOR J=1 TO 20
60 PRINT USING "###Ø#"; A•10;
70 PRINT TAB (25);: PRINT USING "###Ø#";I
80 D=K1•W+A
90 I=I+D
100 A1=K2•(I+K3•D)
110 A=A+K4•(A1–A)
120 NEXT J
140 END
```

Cuando se ejecuta el programa se empieza por determinar cuatro constantes. K1 se refiere a la aerodinámica del coche. K2 y K3 tienen que ver con la percepción del conductor de la señal de error. K4 se refiere al tiempo que necesita el conductor para reaccionar. El programa pide a continuación que se entre la velocidad del viento en kilómetros por hora. Se toman como positivas las ráfagas que soplan de un determinado lado, por ejemplo la izquierda. Para considerar las que soplan por el otro lado en dirección contraria, se introduciría un número negativo.

A continuación el programa imprime una tabla de ángulos y de distancias del coche con respecto a la trayectoria deseada. No se ha intenta-

---

[10] William T. Powers, *Byte*, junio de 1979, página 132.

do un análisis exacto en términos físicos del proceso; los cálculos reales serían mucho más complicados.

Se supone que cada bucle, a medida que J se incrementa, corresponde a un intervalo de tiempo, por ejemplo a 1/10 de segundo. La línea 60 imprime el ángulo de las ruedas multiplicado por 10 para que el número resulte perceptible. La línea 70 imprime la desviación I respecto a la trayectoria deseada. La línea 80 calcula D, la desviación extra que ha tenido lugar desde la última ejecución del bucle. D es igual a una constante, K1, multiplicada por la velocidad del viento, W, más el ángulo de las ruedas, A (que en general tendrá el signo opuesto).

La línea 90 suma la nueva desviación al total anterior I. A continuación, el programa supone que el conductor se ha dado cuenta de lo que ocurre y hace algo al respecto. Por desgracia, existe un cierto retraso entre la operación del ojo, el cerebro y los músculos, lo que hace imposible que el conductor modifique A inmediatamente para enfrentarse a las nuevas condiciones. Para simular esto, el programa calcula un ángulo intermedio A1 en la línea 100. Los estudios realizados acerca de las reacciones humanas demuestran que las reacciones de las personas a este tipo de situaciones son proporcionales (con constante de proporcionalidad K2) a la desviación I y a otro sumando (que depende de la velocidad con que cambia I). Este sumando es D, lo que ha cambiado I desde el último bucle, y está multiplicado por la constante K3.

La línea 110 cambia a continuación A en una cantidad proporcional a la diferencia entre A y A1 (para tomar en cuenta el retraso de actuación de los servos internos del propio conductor) mediante la constante K4. Si hiciésemos actuar este programa con el coche en la trayectoria deseada y sometido a una ráfaga de viento de 48,280 km/h, el comportamiento sería el siguiente: muy rápidamente el coche se desviaría de su trayectoria 0,57 m en dirección del viento. Sin embargo, en el bucle siguiente se girarían las ruedas 19,8 grados en la otra dirección, con lo que el coche se encontraría a unos 15 cm de la trayectoria correcta. Volvería a desviarse pero muy pronto se llegaría a un equilibrio con 0,24 m de desviación.

Este comportamiento es típico de muchos servosistemas. Existe un período inicial de cambios en torno a la posición final, seguido de un estado estable con un error pequeño. Debe existir alguna señal de error, de otro modo no girarían las ruedas en la dirección del viento y nada impediría que el coche se desviase en la dirección del viento hasta coli-

sionar con el tráfico en dirección contraria. También es interesante señalar que el sistema es relativamente insensible a los cambios en las constantes. Pruébese de dar diferentes valores a K4, que corresponde al tiempo de reacción del conductor. Si K4 se fija en 0,4 (rapidez de reacción de un piloto de fórmula 1), la corrección por la ráfaga de viento es instantánea. Si se fija en 0,5, el coche nunca se equilibra sino que continúa dando bandazos de un lado para otro incesantemente. Con K4 = 0,6 el conductor corrige en exceso, las desviaciones del coche se hacen cada vez mayores y pronto acaba en la cuneta o bajo las ruedas de un camión que circula en dirección contraria.

Utilizando un servo bucle, el computador puede permitirse ignorar gran cantidad de hechos respecto al mundo exterior, ya que realiza un pequeño experimento antes de emitir cualquier orden de control. Sólo necesita conocer, antes de empezar, el tipo de fuerza de control que debe utilizar. De lo contrario, podría hacer que las ruedas girasen con tal violencia que el coche volcase.

Los servo bucles se apoyan en el *feedback* (retroacción) negativo que reciben; es decir, en la amplificación de la señal de error entre lo que está ocurriendo y lo que debería ocurrir. Si los sistemas sensores y de control no dispusieran de servo bucles tales como el descrito, que incluye el coche, a la carretera y a la atmósfera, así como sus irregularidades e imperfecciones, sería imposible conseguir que los computadores controlasen máquinas.

William Powers dio un paso hacia delante al proponer una nueva teoría del control, según la cual los servo bucles no examinan lo que ha ocurrido (por ejemplo, donde está el coche), sino simplemente lo que los sensores del sistema dicen acerca de la posición del coche. El objetivo del servo bucle es reducir la entrada de datos acerca de la desviación del coche al valor más pequeño posible.

El aspecto más interesante de los servo bucles es que trabajan sin saber nada acerca del mundo exterior. Pueden imaginarse como cajas negras provistas de dos entradas (la señal de error y la señal de control), una salida, y algunos diales en sus paredes para fijar las constantes. Esta caja (o, más bien, esta subrutina) puede así controlar cualquier cosa. En nuestro ejemplo lo que hemos hecho es conducir un coche, pero el mismo tipo de servo bucle puede controlar también el acelerador. Un pro-

grama para la conducción de automóviles podría utilizar en ambos propósitos la subrutina del servo dando diferentes valores a las constantes.

Pero, además, las rutinas de bajo nivel, tales como éstas, pueden ser controladas por otras de más alto nivel. Por ejemplo, en una carretera desierta se determina la velocidad en función de la distancia del horizonte. Si se tienen 16 km de autopista rectos, puede conducirse a 150 km/h; si la carretera desaparece detrás de una curva, es preferible disminuir la velocidad a 30 km/h. En la práctica, por supuesto, habrá otros vehículos en la carretera, y las distancias a que éstos se encuentran y las velocidades con que circulen también afectarán a la velocidad que se fije el servo de nivel más bajo.

A un nivel aún más elevado, un servo puede controlar el propósito del viaje. Si se emprende un viaje en coche desde Nueva York a Troya para visitar a una tía anciana, mientras la casa de la tía no esté a la vista, la velocidad vendrá determinada por el servo de segundo nivel y su entrada de control es lógica: ‘0’ significa “Todavía no hemos llegado”; ‘1’ significa “Ya hemos llegado”, para el coche.

Al nivel siguiente, un servo podría decir si en realidad queremos emprender este viaje. Tomaría como señal de control diversas normas de “buen” comportamiento, que dicen que hay que visitar a las tías ancianas, e introduciría una señal de error en el sistema si esto no es así. Un servo realmente de alto nivel sería capaz de realizar consideraciones financieras y tomaría como señal de error la diferencia entre la cuenta bancaria de nuestra tía anciana y la nuestra.

EL SALTADOR

La disponibilidad de microcomputadores baratos y potentes permite construir máquinas que de otra manera jamás se habrían podido imaginar. Un ejemplo de éstas es el saltador de una sola pata (*single-leg hopper*[11]), especie de pequeño canguro mecánico que se construyó para estudiar la forma como se mueven los humanos y los animales y también para experimentar las posibilidades de construir vehículos que no vayan sobre ruedas o raíles.

---

[11] Marc H. Raibert e Ivan E. Stherland, *Scientific American,* enero 1983, páginas 32-41.

Los diseñadores de esta máquina pudieron construir, utilizando un microcomputador, un mecanismo que se mantenía en pie independientemente de su forma geométrica. Así, igual que ocurre a un animal, si deja de concentrarse cae. Consiste en un "cuerpo" (un armazón que sostiene varios instrumentos sensores y las conexiones a los computadores) y una "pata" (un gato neumático, de acción rápida y de baja fricción con un pie antideslizante en su extremo).

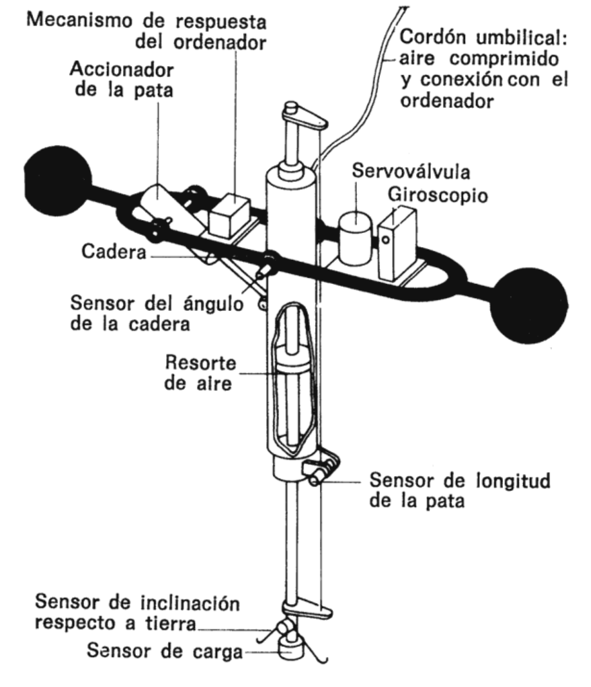


Fig. 26. Detalle de una de las primeras versiones de un saltador, que precisaba de un apoyo lateral para funcionar.

Puesto que el ingenio humano todavía resulta incapaz de igualar la relación potencia-peso del músculo, la máquina debe obtener su energía del exterior a través de una manguera de poco peso por la que circulan aire y aceite comprimidos. La pata está suspendida en medio del cuerpo y puede balancearse hacia delante o hacia atrás o de lado a lado gracias a un

conjunto de gatos. El computador, a su vez recibe información sobre la posición angular de la pata mediante los sensores de longitud acoplados a estos mismos gatos. La fuerza motriz es el gato principal de la pata. Al alargarlo súbitamente se consigue que la máquina salte; al caer comprime de nuevo el gato y rebota. Para compensar la pérdida de energía debida a la fricción, sólo se necesita dejar un poco de aire en el interior del gato y la máquina saltará en el mismo lugar a su frecuencia de resonancia natural.

Los diseñadores se dieron cuenta de que si conseguían hacer saltar a la máquina en el mismo lugar sin que cayese, no resultaría difícil conseguir que se moviese. Para que saltase de forma adecuada necesitaban tres servo bucles (véanse pp. 189-194). El primer bucle controla la altura del salto introduciendo o extrayendo aire del muelle neumático del gato principal. El peso de la máquina y la capacidad del gato forman un sistema resonante. La mayor parte de la energía que se necesita para saltar proviene del rebote anterior. El servo bucle empieza con un sensor de altura situado en el gato principal y va al computador donde se compara con una altura de salto ideal preestablecida. Si es preciso, el computador da instrucciones a las válvulas de aire para introducir o extraer aire en el gato.

El segundo bucle calcula el ángulo adecuado de la pata, mientras la máquina está en el aire, para que aterrice en equilibrio. Para esto se toma en consideración la velocidad de avance de la máquina y la inclinación del cuerpo (hacia delante o de lado). Un programa único de computador funcionará si la máquina está saltando en un mismo punto, si empieza una carrera, si se mueve a velocidad constante, si salta sobre algún obstáculo o si disminuye su velocidad. Este bucle tiene como entradas las señales del giroscopio, que proporcionan información sobre las aceleraciones angulares y la posición relativa entre gato y cuerpo. Si el saltador salta en un mismo lugar, este bucle corregirá los pequeños desequilibrios del cuerpo cuando éste se inclina hacia uno u otro lado. Si la máquina empieza a correr hacia la izquierda, se mueve la pata hacia la derecha para que todo el saltador se incline hacia la izquierda de manera que el próximo salto lo desplace hacia la izquierda al mismo tiempo que hacia arriba. El tercer bucle estabiliza el aparato cuando está en el suelo.

Cuando estos tres bucles funcionan adecuadamente, la máquina imita muy bien la carrera de un canguro. El saltador balancea su pata hacia

delante y hacia atrás del mismo modo como lo hace el animal; debe hacerlo así para mantenerse en equilibrio en el aire. Los diseñadores del saltador intentan ampliar su máquina para hacer una de cuatro (o seis) patas, que podría ser de gran utilidad en terrenos demasiado agrestes para ruedas u *hovercraft.* Sin embargo, tal como vimos en la página 129, la visión del computador todavía no es lo bastante buena como para proporcionar "ojos" al animal mecánico. Necesitaría un conductor humano para indicarle dónde poner sus patas.

ROBOTS

Aunque ninguno de nosotros ha visto nunca moverse un robot metálico, como el R2D2 de *La guerra de las galaxias,* todos sabemos perfectamente cómo debería comportarse. Cualquier obra de ciencia ficción que se precie tiene gran cantidad de robots: robots pedantes y correctos, robots patéticamente rotos, robots espías que pretenden ser pedantes y robots locos que deben enviarse al triturador de chatarra porque han olvidado las "Tres Leyes de la Robótica" de Asimov.

Como veremos más adelante, la ingeniería que comporta un robot de metal clásico está muy alejada de nuestras posibilidades actuales. Nadie tiene la menor idea de cómo ponerse a construir un robot que parezca un ser humano: un androide similar a los que aparecían en *Blade Runner.* Lo que sí todos hemos visto son robots acoplados a otros tipos de maquinaria.

Una vez que hemos conseguido que el computador reciba señales del mundo real gracias a los sensores, podemos hacer que realice muchas tareas que en caso contrario deberían ser realizadas por personas. Esto no es nada nuevo. Desde los comienzos de la era industrial, las máquinas de vapor necesitaban un control continuo del suministro de vapor al cilindro: cuando el pistón está en la parte superior de su carrera, debe entrar vapor en la caldera; cuando está en la parte inferior, el vapor utilizado tiene que escapar a la atmósfera. Los fabricantes proporcionaban dos grifos para realizar estas operaciones y el operador debía contratar un muchacho que los abriese y cerrase en los momentos correctos.

Según una leyenda, uno de estos muchachos se dio cuenta de que podía improvisar un sistema de cuerdas que hiciesen el trabajo automáticamente y así echarse tranquilamente a dormir después de comer.

Así, sin proponérselo, este muchacho había construido el primer robot: un robot-válvula de abrir y cerrar. Este robot presentaba los tres problemas de la robótica con los que todavía nos enfrentamos hoy en día: detección, inteligencia y ejecución.

El robot tiene que detectar señales provenientes del mundo real y saber qué hacer con ellas. En este caso tenía que detectar cuándo debía abrir y cerrar las dos válvulas. A continuación tenía que aplicar la fuerza necesaria para hacer lo que se quería: girar las válvulas. Este primitivo robot hacía las dos funciones en una sola, ya que el balancín del motor tenía tanta fuerza que las cuerdas sujetas a él en los lugares adecuados podían fácilmente hacer girar las válvulas.

En muchos casos la captación de señales y su ejecución son funciones completamente separadas; hoy en día, entre ambas se encuentra a menudo un microprocesador.

ROBOTS EN LA INDUSTRIA

En las páginas 184-189 vimos la manera como los computadores pueden recibir información procedente del mundo real: a partir de una gran cantidad de sensores diferentes, desde un simple interruptor de contacto colocado en una ventana, hasta una cadena de radares de ámbito nacional. Una vez obtenida la entrada, el computador tiene que calcular lo que se desea. Puesto que, tal como vimos en las páginas 174 y 178, todavía no se sabe cómo lograr que los computadores vean y oigan de la forma que nosotros lo hacemos, una parte crucial del arte de la robótica es la obtención de formas de entrada que el computador pueda interpretar.

Si un robot hace funcionar un tren de laminación en una acería, no resulta en cambio suficiente que una cámara de televisión enfoque el acero incandescente que sale del tren y espere que el computador deduzca el grosor de la plancha de acero a partir de esta imagen. Hay que instalar dispositivos que midan el grosor de la plancha con la precisión adecuada y proporcionen los resultados dispuestos de forma conveniente a la interface del computador.

Una vez preparado el computador para que interprete la entrada procedente de este dispositivo como grosor del acero, seguirá interpretándola del mismo modo aunque ésta provenga, por ejemplo, de un termóme-

tro. Después de haber medido el grosor del acero, el computador toma una decisión y actúa en consecuencia accionando unos enormes gatos de tornillo que modifican la distancia entre los rodillos. El sistema en su totalidad constituye un perfecto robot, que puede reemplazar al “Viejo Pedro”, el trabajador especializado que acostumbraba a ajustar los rodillos a mano. Pero este robot no se parece en nada al que imaginábamos.

En la industria moderna hay una gran cantidad de robots de este tipo, muchos de ellos especializados y capaces de adaptarse a una pequeña gama de funciones. El robot del tren de laminación, por ejemplo, probablemente sólo puede ser adaptado a la producción de planchas de diferentes grosores. Si quisiéramos cambiar esta función, por ejemplo para fabricar raíles de ferrocarril, casi con toda seguridad tendríamos que solicitar la ayuda de un equipo de ingenieros para que diseñase de nuevo todo el sistema. Su flexibilidad no tiene punto de comparación con la del trabajador a quien reemplazó. Pero, por otra parte, tampoco tiene ninguna de las predisposiciones humanas al error, equivocaciones, fatiga y malhumor (ni sus costos).

El reto al que se enfrenta el diseñador de robots consiste en producir una máquina que sea lo suficientemente especializada para que realice perfectamente su trabajo y, al mismo tiempo, lo bastante flexible para que pueda ser reprogramada sin que sea necesario diseñarla de nuevo. Este problema resulta especialmente acuciante en los robots dotados de visión. Los robots para las cadenas de producción, que deben “ver” la siguiente pieza del montaje en el momento que llega, han de estar provistos de sensores muy especializados, tales como rayos de luz que la pieza interrumpe al llegar. Si se cambia la forma de la pieza, los sensores deben colocarse en una nueva posición y la máquina tiene que ser reprogramada. Otro problema es que todavía no somos capaces de construir mecanismos tan sensibles y versátiles como la mano humana. A un robot puede proporcionársele una pistola de *spray*, un cabezal de soldadura o un taladrador; sin embargo, no puede construirse un dispositivo que sirva para las tres operaciones.

De todas maneras, los robots empiezan a reemplazar a los trabajadores en las tareas pesadas y repetitivas o en las que deben realizarse en condiciones difíciles o peligrosas. Un lugar perfecto para un robot es la sección de soldadura en las cadenas de fabricación de automóviles y allí justamente es donde se encuentran. También se empiezan a introducir en

las minas de carbón, liberando a los hombres de un trabajo inadecuado para el cuerpo humano.

¿CÓMO FUNCIONA UN ROBOT?

Cualquier trabajador competente podría montar una máquina si se le proporcionasen las piezas necesarias y un dibujo del montaje. Pero consideremos las enormes complicaciones a las que nos enfrentamos si tratamos de construir un robot que realice las mismas operaciones. Debería tener un sistema de visión que le permitiera identificar una pieza determinada de entre un montón de ellas. Además, necesitaría un sistema experto inteligente que fuera capaz de deducir, a partir de los dibujos, el orden en que deberían montarse las piezas. Por último, debería fabricarse una “mano” que pudiese manejar todo tipo de herramientas; que tuviese la delicadeza necesaria para montar un mecanismo de relojería y la fuerza suficiente para triturar rocas.

La máquina debería ser en conjunto más barata que un trabajador y tan fácil de reprogramar como él. Un especialista gana alrededor de 15.000 libras en Gran Bretaña o 30.000 dólares en Estados Unidos, lo que representa los intereses y la amortización durante cinco años de una máquina que cueste 85.000 dólares, más o menos lo que vale la pena pagar por un robot. Sin duda, esta cantidad no es tan fácil de calcular, porque quizá sea interesante pagar más por una máquina que trabaje más horas que una persona o que sea capaz de realizar trabajos más duros o más peligrosos de los que puede hacer un ser humano. Sin embargo, se invierte muy poco en vistas a la producción de un hombre sintético. Utilizando la tecnología existente en la actualidad, la construcción de un robot humanoide para todo tipo de necesidades, se encuentra fuera de nuestro alcance. Lo que actualmente denominamos “robot” no es más que un periférico de computador que puede coger cosas o manejar una herramienta en una cadena de montaje.

Un robot consiste en un brazo único, cuyo extremo puede moverse en cualquier dirección en un radio de cerca de 3 m. En el extremo del brazo hay una juntura articulada que puede girar e inclinarse hacia arriba, hacia abajo y hacia los lados. En lugar de una mano multiuso el robot acostumbra a estar dotado de herramientas especializadas (un soplete, un aspirador o una pistola para pintar).

La tarea del programador consiste en la realización de un lenguaje que permita al usuario del brazo-robot programarlo fácilmente. En una cadena de montaje, por ejemplo, el usuario debe poder ordenar al robot que espere la llegada de la carrocería del coche. Cuando se interrumpe el haz luminoso de una célula fotoeléctrica, el robot recibe la señal de que la carrocería está en la posición adecuada y, a continuación, la orden de mover la mano hasta el principio de la costura del techo y soldarla. Esto requiere instrucciones tales como: "Ir desde el Estacionamiento hasta el Inicio-costura. Desplazarse hasta el Final-costura, soldando al mismo tiempo. Ir al Estacionamiento".

"Soldar" es una subrutina que detecta la distancia existente entre el cabezal soldador y la carrocería del coche, controla la circulación de la corriente y la longitud de la vara de soldar y hace todas las cosas pertinentes. "Inicio-costura" y "Final-costura" son posiciones en el espacio que están preprogramadas en el computador a partir del diseño de la carrocería del coche. El programador del robot puede introducir estas dos posiciones como coordenadas o colocar el robot en posición de "aprender" y mover manualmente el cabezal de soldadura. El lenguaje debería tener rutinas que suavicen las irregularidades de este tipo de enseñanza manual.

Debería ser factible que el robot procesara cualquier tipo de subrutina: pequeños conjuntos de movimientos que el programador quiere que sean realizados en distintos lugares. Supongamos que se utiliza el robot para poner en una caja botellas a las que luego se les enroscan los tapones. El movimiento necesario para enroscar cada tapón en una botella es el mismo cada vez, pero debe realizarse en un lugar distinto.

Cuando los sensores sean más sofisticados, los robots podrán programarse de forma más general, dándoles el mismo tipo de instrucciones que se darían a una persona de pocas luces. «Coger las botellas de la línea de producción, enroscar un tapón en cada una de ellas y llenar la caja».

ROBOTS DE ADIESTRAMIENTO

Un periférico interesante para un microcomputador, aunque bastante especial, es el robot de adiestramiento: una versión pequeña, que puede ponerse encima de una mesa, de los monstruos que se encuentran en las

fábricas. De hecho, una de las funciones de los microcomputadores es la popularización de la informática, por lo que estos robots no son tan inútiles como a primera vista pudiera parecer. Además, para una empresa que maneje gran cantidad de objetos pequeños, estas máquinas serían de gran utilidad, ya que se podría incluso construir una cadena de producción.

Una de estas máquinas está fabricada por la empresa Mitsubishi de Tokio. Se vende con su propio microcomputador CP/M, pero puede conectarse fácilmente a cualquier máquina que disponga de una salida serial.

El brazo del robot tiene seis pequeños motores que lo mueven. Estos motores, lo mismo que en muchos robots industriales, son motores de escalón accionados por series de pulsaciones eléctricas que los hacen girar una pequeña distancia fija por pulsación. La salida del motor está adaptada a un engranaje, de modo que las extremidades del robot se mueven entre 0,04 y 0,08 grados por pulsación. Esto permite que el computador de control "sepa" dónde se encuentra cada articulación contando simplemente las pulsaciones que se le envían. Sin esta característica el robot debería tener un conjunto de sensores, caro y complicado, que midiese e informase de la posición de sus articulaciones.

Un motor situado en la base hace girar todo el robot alrededor de un eje vertical. Las articulaciones del hombro, codo y muñeca funcionan horizontalmente, de manera que las pinzas pueden situarse en cualquier punto de una semiesfera trazada alrededor de la máquina. La muñeca puede dar vueltas sobre sí misma una y otra vez, lo que no puede hacer una muñeca humana. Un sexto motor cierra las pinzas estirando un cable.

Las instrucciones se envían al robot del mismo modo como se hace con una impresora: como una serie de caracteres. Por ejemplo, la línea de BASIC:

LPRINT "H"

hará, cuando la máquina esté conectada al conector de salida de la impresora, que el robot vaya a "Home" —quede plegado sobre sí mismo— con todos los motores colocados en el punto final de su recorrido. El micro computador de control sabe de qué punto partieron y puede asimismo saber dónde se encuentra en cada momento durante la operación.

El microcomputador de la base se programa teniendo en cuenta el tiempo necesario para acelerar y frenar las articulaciones, de modo que todos los movimientos se realicen suavemente y con absoluta precisión.

La máquina responde a quince órdenes distintas, que se presentan en dos niveles. Al nivel más bajo se hace que cada articulación gire un determinado número de pasos. Esto se ordena a través del teclado o mediante un panel de mandos de forma que la máquina pueda aprender. Por ejemplo, al comienzo del programa del juego de “tres en raya” la máquina extendería el brazo para indicar dónde supone que debe estar el centro del tablero y dónde buscará las pilas de fichas. El operador tendrá que ajustar la posición del robot y de las piezas del juego de modo que la máquina ciega pueda encontrarlas durante la partida.

Una vez establecida una posición (por ejemplo, sobre la pila de fichas blancas), se le puede dar un número que queda almacenado en el RAM del robot. Así, mientras esté conectado a la red, la máquina recordará su posición e irá hacia la pila respondiendo a la orden.

```
LPRINT “M3”
```

Una vez en la posición adecuada, puede cerrar las pinzas para coger una pieza y moverse a otra posición donde la deja. No resultaría demasiado difícil escribir un programa que hiciese que el brazo cogiese pequeños tubos, suministrados en una cinta, y los colocara en cajas. Las desviaciones necesarias para poner cada tubo en su posición apropiada dentro de la caja se incorporan en el programa. El programa del computador de control puede enseñar al robot a hacer una tarea determinada enviándole una lista completa de posiciones en forma de números específicos de pasos para cada motor, contados a partir de la posición “nido”. Por ejemplo, el programa para jugar a “tres en raya”, se inicia dando una serie de posiciones:

```
10 LPRINT "P1,0,372,–958,592,–592,0"
20 LPRINT "P5,739,–707,–431,–86,–1114,0"
```

y así sucesivamente. Las pinzas pueden desplazarse a estas posiciones, durante la ejecución del programa, sin más que especificar el número correspondiente.

ANDROIDES

La idea de una máquina que se comporte como un ser humano (un androide) resulta fascinante. Si tenemos en cuenta que cualquier computador imita y mejora los procesos mentales del hombre, podemos considerarlo una especie de androide. Pero, sin duda, no parece humano y carece de las facultades que permiten al hombre moverse, manipular objetos, ver, oír y sentir. La esperanza de crear una máquina de este tipo, o al menos gran parte de ella, ha inspirado a muchos investigadores importantes en el campo de la ingeniería y de la informática, y también en el de las artes. De hecho, los constructores de androides que han cosechado mayores éxitos se encuentran en la industria cinematográfica (como en la película *Blade Runner)* donde pueden utilizarse seres humanos para accionarlos. Se han realizado muchos esfuerzos en este campo, pero finalmente tenemos que reconocer que, comparados con la Madre Naturaleza, sabemos muy poco sobre informática e ingeniería. Vamos a tratar a continuación por separado los principales problemas que presenta la creación de androides.

Necesitamos una máquina capaz de moverse por sí sola durante varios días, a través de terrenos agrestes, subiendo escaleras o árboles e, incluso, acantilados, antes de que sus baterías se agoten. Debería ser capaz de coger un peso equivalente al suyo, transportarlo, y con las mismas manos y brazos coger y enhebrar una aguja. En la actualidad no tenemos ninguna máquina que ni remotamente realice estas operaciones. Una máquina que funcionase a base de motores eléctricos y baterías dejaría de hacerlo al cabo de una hora de moverse por una superficie llana; un simple tramo de escaleras agotaría toda su energía. Un brazo mecánico suficientemente fuerte para competir tirando de una cuerda debería pesar cerca de cincuenta kilos. Incluso aunque supiéramos cómo construir piernas para andar (lo que no podemos hacer todavía), su peso se acercaría más a la tonelada que a las decenas de kilos.

El ojo humano tiene el equivalente de unos 3 millones de pixeles, mientras que las mejores televisiones tienen sólo 1 millón. Pero incluso si tuviéramos un ojo mecánico lo bastante sensible, no podríamos procesar su información en menos de varias horas (en cambio, el ojo y el cerebro lo hacen en 1/25 de segundo) y todavía no sabemos hacer más del 1% del procesamiento necesario.

El cerebro humano contiene cerca de 10.000 millones de neuronas. Cada neurona está conectada a muchas otras y tienen una capacidad de almacenamiento de bytes desconocida, aunque podemos suponer que puede almacenar 100 bytes. En este caso, el cerebro equivale a un billón de bytes, es decir, el contenido de un cubo de 2,8 m^3 lleno de chips de memoria actuales. Y esto suponiendo que supiéramos organizar la memoria en caso de tenerla. Un microcomputador actual de 16 bits tardaría más de tres semanas en buscar una palabra de cuatro letras en esta memoria.

Por esta razón, debemos aceptar que los androides de ciencia ficción se encuentran en un futuro muy lejano. Curiosamente, hace diez o más años se realizaron serios intentos de construir un androide. Los diseñadores utilizaron lo que hoy considerarían computadores excesivamente grandes, pero esto no importa demasiado. Lo que importa es el software y éste no ha cambiado mucho: aquellos experimentos iniciales demostraron que existía tal diferencia entre las cualidades de las máquinas y las de los humanos que no había ninguna posibilidad real de superarla en aquella etapa del desarrollo de la informática. Los androides se llamaron Shakey, construido en la Universidad de Stanford, y Freddy, construido en la Universidad de Edimburgo.

Shakey era un robot móvil, dotado de un brazo, pinzas y una cámara de televisión, que rodaba de un modo poco estable por un pequeño mundo de cinco habitaciones, una rampa y varias cajas que podía manipular. Su computador era estático y estaba conectado al androide por cable. Freddy era un dispositivo estático, con una pinza colocada en un brazo suspendido del techo. Miraba a los objetos esparcidos sobre una mesa situada debajo, que podía moverse en dos direcciones accionada por motores eléctricos.

Freddy fue un intento de combinar sensores visuales, inteligencia de máquina y el brazo de un robot para formar un trabajador mecánico de una cadena de montaje. Aunque hace algo más de una década que Freddy fue desmantelado, probablemente los primeros robots inteligentes funcionarán de manera muy parecida a como lo hacía Freddy. Fue programado para montar pequeños juguetes, formados por piezas de madera, apilados sobre su mesa de trabajo. Antes de cada ejecución, el operador humano tenía que realizar dos conjuntos de ejercicios de preparación. Primero, debía mostrar a Freddy todas las partes del juguete en todas las

posiciones en que podían encontrarse sobre la superficie plana de trabajo. Luego, tenía que decir al robot cómo coger con las pinzas cada una de las partes y cómo colocarlas en posición de montaje. Finalmente, tenía que programar la máquina para que montase las distintas partes en el orden adecuado. Una vez realizadas todas estas cosas, Freddy se comportaba con una inteligencia notable.

Trataba de coger los elementos que necesitaba de la pila de piezas y los colocaba aparte en las posiciones estandarizadas en que debían estar antes de empezar el montaje. Si no podían verse piezas individuales como entidades separadas, Freddy atacaba al montón de piezas e intentaba sacar algunas a unidades. Si esto no daba resultado, Freddy balanceaba su brazo golpeando la pila en un intento de desmontarla.

Como hemos visto anteriormente, los robots tienen mucha fuerza pero muy poco cerebro. Tampoco pueden hacer muchas cosas para sentir o percibir. Por ejemplo, un robot soldador en una cadena de montaje de automóviles recibirá la señal de que ha llegado un nuevo chasis mediante el cierre de un interruptor. Entonces, se pone a soldar, como un ciego, en el lugar donde se le ha indicado, y si el coche no está donde debería estar, el robot trabaja en el vacío.

Este caso es bastante ilustrativo, ya que la descripción de la situación final contiene algunas claves evidentes que indican por dónde empezar. Sin embargo, en la práctica lo que se desea es obtener respuestas a preguntas tales como: «¿Cómo puedo llegar a ser rico y famoso?» Si la pregunta tuviera algunos indicios que permitieran conocer la respuesta, no trataríamos de obtenerla del computador.

Vimos en las páginas 174-178 que resulta imposible imitar la visión humana. Incluso esquemas más sencillos no han tenido mucho éxito. Una cosa es hacer que una máquina pueda reconocer unas cuantas piezas de madera en el laboratorio, donde las condiciones son controlables, y otra muy distinta producir un equipo que pueda instalarse en cualquier fábrica y que funcione con fiabilidad. En estas circunstancias, el sistema de visión se encuentra con vibraciones, ruidos, polvo, suciedad y una iluminación difícil de prever. Una sombra inesperada, una mancha de aceite o el reflejo de una máquina pueden hacer que el objeto aparezca completamente diferente en un sistema de visión demasiado simple.

Incluso con una perfecta iluminación resulta muy difícil descubrir qué es lo que la cámara está observando. Un sistema sencillo pero eficaz

consiste en iluminar los objetos con una franja muy delgada de luz que enfoque oblicuamente la escena que divisa la cámara. Para comprobar que el objeto que el robot debe manipular se encuentra en la posición adecuada, el sistema de visión tiene que comparar la franja brillante que ve la cámara con una versión anterior introducida en su memoria.

Acabamos de decir con cierta ligereza que el computador sólo tiene que comparar, a medida que el objeto pasa a través del rayo luminoso, la forma de la franja de luz con la versión que posee en su memoria. Sin embargo, esto no resulta nada fácil, especialmente cuando el objeto está orientado al azar y los programadores se ven obligados a reducir su descripción en el computador a un conjunto de números aparentemente irrelevantes: como, por ejemplo, la razón entre su longitud y su circunferencia.

También el tacto exige mucho tiempo de procesamiento. No es excesivamente difícil dotar a las pinzas del robot con interruptores de contacto o almohadillas de presión que indique si las pinzas han cogido algo y, en caso afirmativo, la presión con que lo han hecho. Pero incluso para algo tan sencillo como esto, la capacidad de procesamiento requerido es inmensa. Unos cuantos sensores necesitan la continua atención del computador si no se quieren perder las sacudidas transitorias que podrían indicar alguna información vital del tipo: «Intenté coger la pieza, pero se me cayó de la mano». Si esto se realiza con un solo computador, la cadena de producción tendrá que ir a paso de tortuga; si se utilizan varios, el diseñador se enfrentara a los problemas todavía no solucionados del procesamiento en paralelo (véase p. 245).

Todos los robots industriales deben instruirse sobre el modo de realizar su trabajo, como se hizo con Freddy. El hombre debe encontrar una estrategia adecuada y programársela: «Primero haces esto y después aquello...» Si un día los robots tienen que cumplir las esperanzas que sobre sus posibilidades tenemos en la actualidad, deberán tener la capacidad de descubrir por sí mismos lo que han de hacer en cada momento.

Shakey era, entre muchas otras cosas, un experimento en la toma de decisiones. Shakey se desplazaba por su pequeño mundo de habitaciones, puertas y cajas que podía empujar de un sitio para otro. Para facilitarle las cosas, se le hacían hacer tareas del tipo “Colocar la caja de la habitación 3 cerca de la caja de la habitación 4 en la habitación 2”. Primero, Shakey tenía que explorar su mundo para saber dónde se encontraban las

cajas. A continuación, el programa del computador (llamado STRIPS) tenía que encontrar los pasos necesarios para producir el resultado deseado. Para esto disponía de varias acciones que Shakey podía ejecutar. Podía desplazarse por sí mismo, también podía aproximarse a las cajas, podía empujarlas y podía atravesar las puertas.

Resulta fácil ver que existen dos modos de hacer esta operación:

Empujar la caja de la habitación 3 a la habitación 2, luego ir a buscar la caja de la habitación 4

Empujar la caja de la habitación 4 hasta la habitación 2, luego ir a buscar la caja de la habitación 3.

Para empezar, si sabemos dónde se encuentra la caja de la habitación 4, no resulta difícil descubrir que debería empujarse a la habitación 5 y luego a la habitación 2. Y así sucesivamente. Pero incluso en este caso deben hacerse muchas operaciones de cálculo informático. El problema básico con que nos encontramos es que en cada etapa existen varias cosas que el computador puede hacer a continuación. Empieza en la habitación 5: ¿Qué debería hacer? Puede permanecer en el mismo sitio o ir a las habitaciones 1, 2, 3 o 4. En la habitación 4 puede empujar la caja a otro lugar de la misma habitación o sacarla por la puerta. Una vez que la caja de la habitación 3 está en la habitación 5, puede empujarla a las habitaciones 1, 2, 3, 4 o dejarla en la 5... Rápidamente resulta demasiado complicado para explicarlo con palabras, de modo que es mejor dibujar un diagrama como el de la figura 27.

Pero incluso este diagrama está lejos de ser completo. Cualquiera que trate de dibujar todas las cosas que Shakey puede hacer antes que mover todas las cajas a todas las habitaciones, habrá tenido que usar casi tres hectáreas de papel. Además, este diagrama también nos engaña porque ignora gran cantidad de decisiones que pueden tomarse acerca del modo de empujar las cajas. Así, cualquier programa que avance simplemente (aunque sólo sea en la imaginación) a través de cada paso que puede hacerse desde otro paso, se verá desbordado rápidamente por una «explosión combinatoria» (véase p. 249). Tendrá que hacerse mejor.

Incluso cuestiones tan sencillas como “¿Cómo iré al aeropuerto para coger el vuelo hacia Chicago de las 9?” requieren complicadas deducciones a partir de gran cantidad de conocimientos sobre coches, bicicletas,

taxis, autobuses, trenes, líneas aéreas y sobre dónde es mejor y más agradable esperar.

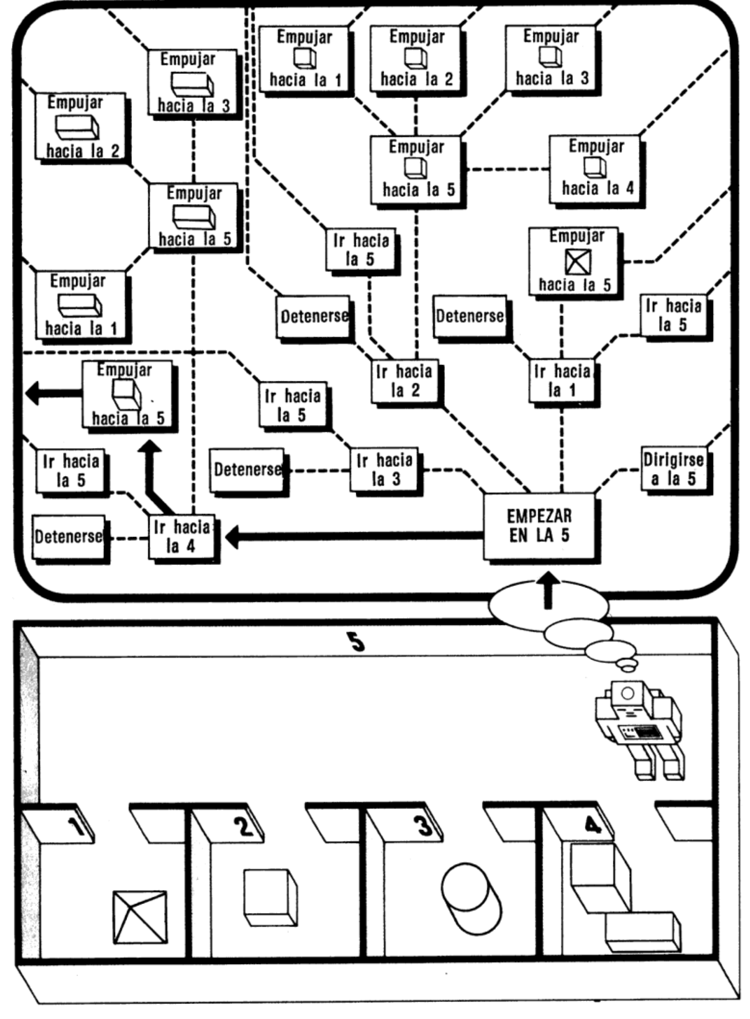


Fig. 27

Hay algo en el cerebro humano que lo hace especialmente adecuado para estas cosas; de no ser así, la raza humana habría desaparecido en el estómago de un animal depredador hace mucho tiempo. Pero los computadores no sirven para esto.

Sin embargo, muchas de las dificultades que se encuentran en este campo provienen de la excesiva ambición de los diseñadores de computadores. Si no olvidamos en ningún momento que un computador no es más que una complicada máquina de escribir eléctrica que puede hacer gran cantidad de trabajos aburridos y peligrosos, fácilmente veremos el gran avance que supone.

REDES

Todo negocio (y hasta podría decirse que gran parte de nuestra civilización) se basa en el envío de mensajes. Se hacen llamadas telefónicas, se envían cartas, programas de TV, facturas, cuentas, informes, libros, discos y se dejan notas encima de la mesa de la cocina. En la actualidad, se emplean docenas de tecnologías distintas para transmitir estos mensajes, desde los simples papeles escritos a mano hasta los radiosatélites.

A medida que los computadores se introducen en nuestra vida cotidiana, aumentan las posibilidades de conectarlos a redes: desde las muy próximas al punto de donde proviene el flujo de datos, como por ejemplo en un edificio de oficinas, hasta las de larga distancia, que transmiten reducidos flujos de datos a través de medio mundo.

Redes multiusuario

La posibilidad de unir los computadores personales entre sí permite pensar rápidamente en la creación de una oficina electrónica. Hace años que se habla de esta posibilidad que ahora empieza a ser una realidad en algunos lugares. Sin embargo, antes de entrar en este tema, vamos a ver las formas como pueden unirse los computadores entre sí.

En un sentido muy amplio, existen tres niveles de unión. Las máquinas pueden enviarse mensajes mutuamente a través de líneas seriales rápidas (véanse pp. 28-31); pueden compartir el mismo disco, intercambiando datos de un lado para otro en los archivos, a las velocidades nor-

males de acceso al disco y, finalmente, pueden compartir sus procesadores y memorias.

La utilidad de la primera posibilidad está muy limitada por la velocidad de las líneas. La única justificación de un computador es que haga las tareas a mayor velocidad que un operador humano; si no puede hacerlo, no existe ninguna razón para complicarnos la vida utilizándolo. Las líneas telefónicas rápidas transmiten una pantalla entera de datos en cerca de 10 segundos, lo que resulta suficiente para mensajes cortos como facturas o reservas de hotel, pero no para editar el archivo de un texto del tamaño de un libro.

Este nivel de interconexión es el adecuado para muchas de las transacciones rutinarias entre empresas, en las que los mensajes son cortos y formales; pero no es lo bastante rico para las necesidades internas de una empresa, en la que los empleados pueden enviarse uno a otro memorándums, informes, cartas, proyecciones de tesorería o listas de facturas. En estos casos la electrónica sólo puede ayudar parcialmente; no puede cambiar los fundamentos de lo que ocurre.

Aquí actúa de nuevo la ley de Zipf (véase p. 123), que dice, simplificando, que el volumen de información que necesitan intercambiar dos personas es inversamente proporcional a la distancia entre ellas. Esta ley resulta muy conveniente, ya que los costos de comunicación aumentan rápidamente con la distancia.

El segundo nivel (compartir los archivos del disco) significa que se puede utilizar la electrónica de forma análoga a como lo hace un archivador compartido. Esta es la manera como realmente trabajan las oficinas hoy en día: cada persona realiza su trabajo mediante la selección de un documento, que se procesa de algún modo y se coloca de nuevo en el archivo, para que otras personas lo puedan utilizar cuando lo necesiten.

Evidentemente, una vez informatizado, el sistema ofrece dos grandes ventajas sobre su antecesor. En primer lugar, todo el mundo puede ver instantáneamente lo que ocurre. Por ejemplo, en la oficina de un agente teatral, si todos los ejecutivos utilizasen terminales en un sistema de archivo compartido, cada uno de ellos podría saber al instante en qué situación se encuentra uno cualquiera de sus clientes. Sin un sistema de este tipo, deberían usarse continuamente notas explicativas de uno a otro. En segundo lugar, todos ellos podrían acceder a la información en la base de datos solicitándola de varias formas distintas. «Encontrar un enano

pelirrojo, con una sola pierna, que hable francés» no sería ningún problema con un gestor de base de datos bien diseñado.

En una instalación de este tipo, el usuario individual de cada micro computador debe tener acceso a un disco compartido en el que pueda encontrar todos los archivos informáticos que necesite. En el sistema clásico, dos personas no pueden trabajar independientemente en el mismo archivo al mismo tiempo, ya que la que ha llegado primero ha sacado la ficha del fichero. El computador precisa un equivalente electrónico de esta situación.

Normalmente se consigue bloqueando los records individuales en la base de datos (véanse pp. 150-156) cuando el primer usuario los coge. Si, entonces, otra persona quiere sacarlo, recibe un mensaje en su pantalla que dice algo así como "Record bloqueado", de manera que ya sabe que alguien se le ha adelantado.

Entre los profesionales este problema se conoce con varios nombres: "disputas de archivo" o "choques de records". Es esencial que el software que se utiliza en un sistema de red separe los choques, ya que en caso contrario pueden ocurrir fallos terribles cuando una persona altera un record determinado sin saber que otra también lo está haciendo. Una solución sencilla es utilizar un software de usuario único, gestionado por una base de datos multiusuario tal como Superfile, que trata los choques automáticamente. Este esquema, que permite a varios microcomputadores acceder a los mismos archivos de disco, funciona bastante bien y parece muy interesante respecto al futuro.

Hay esencialmente tres formas de compartir. En un sistema multiusuario, varias personas comparten el mismo procesador. En un sistema multiprocesador, cada persona tiene su propio computador, pero éstos están íntimamente unidos a un sistema central que controla el disco. En un "anillo", los computadores individuales están conectados mediante una línea de alta capacidad, de manera que pueden intercambiar datos de un lado a otro.

En el primer esquema, cada usuario tiene un trozo de memoria y el procesador atiende a cada uno por turnos. Este es el modo como las unidades centrales realizan el procesamiento multiusuario; esto era lógico en el pasado porque los procesadores acostumbraban a ser tan caros que resultaba imposible proporcionar uno a cada usuario; por otra parte, te-

nían la potencia suficiente para llegar a un gran número de usuarios lo bastante rápido para proporcionar a cada uno un servicio satisfactorio.

El principal sistema operativo multiusuario de 8 bits es un derivado de CP/M llamado MP/M. Utiliza *bank-switched memory* para dar a cada usuario 48 K de espacio donde ejecutar sus programas, lo que de por sí ya es una seria limitación. El procesador tiene que desviarse de banco a banco, haciendo el procesamiento que debe hacer para cada usuario. Mientras que un procesador de 8 bits como Z80 es lo bastante potente para la mayoría de los trabajos de oficina, no lo es para servir a media docena de ellos. El rendimiento se "degrada" si utilizan el sistema más de dos personas.

Las máquinas de 16 bits, en particular las que utilizan el procesador 68000 y sus derivados mayores, realizan mucho mejor esta tarea. Son lo bastante potentes para trabajar de la misma forma que los minicomputadores y los main-frames, utilizando a menudo el mismo sistema operativo Unix. El mismo procesador puede servir a varios usuarios y cada uno de éstos podría, probablemente, ejecutar varios programas distintos al mismo tiempo. Esto se llama "multitarea" y se proporciona con el sistema operativo "Concurrent CP/M" en las máquinas de un solo usuario de 16 bits.

Sin embargo, en las máquinas de 8 bits se consigue un esquema mucho mejor dando a cada persona un microcomputador con 64 K completos de memoria, una pantalla y un teclado, y dejando que todos compartan los mismos discos. Este sistema se denomina normalmente "multiprocesador". En la práctica los usuarios acceden a los discos a través de un microcomputador central, cuya única misión es la de servir a los discos y a los operadores; a menudo se le llama "servidor de archivos". La unidad central, que puede ser una máquina de 16 bits, ejecuta algún tipo de sistema operativo multiprocesador. Uno de los más conocidos es el CP/Net; cada usuario piensa que tiene un CP/M. También hay el Turbodos, McNos, Hi-Net y muchos otros. Existen, de nuevo, dos formas distintas de hacerlo. Una consiste en poner los distintos computadores en un bastidor que contenga también el disco y la unidad central proporcionando a cada usuario un terminal. La otra, en dar a cada uno un computador completo, provisto de líneas de alta velocidad que lo unan con la unidad central y el disco.

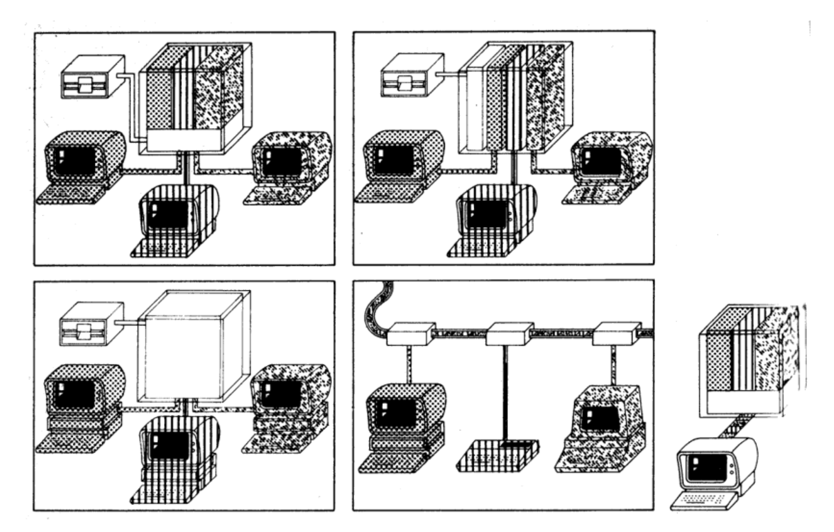


Fig. 28. Hay diversas modalidades para que los computadores puedan ser utilizados por varias personas al mismo tiempo. Una sola máquina (arriba a la izquierda) puede atender los discos y todas las terminales de los usuarios, aparentemente de forma simultánea. De hecho, lo trata todo por turnos, pero a tan alta velocidad que las esperas no se notan. Esto se denomina «multitareas», y es el modo como funcionan los main-frames, los minicomputadores y las máquinas de 16 bits más potentes.
Otro esquema (arriba a la derecha), que se encuentra a menudo en los sistemas de 8 bits, permite que cada usuario acceda a su propio computador a través del terminal. Una máquina suplementaria controla los discos, permitiendo que cada usuario acceda a ellos cuando lo necesite. Este sistema se denomina multiprocesador. Otra alternativa consiste en que los procesadores individuales estén en las terminales de los usuarios y conecten con el «servidor de archivos» mediante un cable de alta velocidad (abajo a la izquierda). Una cuarta variante (abajo a la derecha) consiste en unir los computadores separados (que pueden ser de distintas marcas) a través de pequeños interfaces y de un «anillo» de un cable especial de fibra óptica. Esta solución se denomina a menudo «red de área local» *(local area network;* LAN). Finalmente, un solo computador puede ejecutar varios programas al mismo tiempo para un usuario único (abajo). Esto se llama «procesamiento concurrente».

La principal ventaja del primer método es que se pueden tener líneas lentas en las terminales de los usuarios, ya que sólo de vez en cuando resulta necesario transmitir una pantalla completa de datos. De esta manera, las líneas pueden ser baratas y relativamente largas: hasta varios

centenares de metros. La principal desventaja es que cada usuario debe poseer un terminal separado para visualizar y aceptar los datos. Este terminal podría haber formado parte del procesador del usuario a un coste mucho menor.

El otro método consiste en poner el procesador de cada usuario en su terminal, para darle un microcomputador completo que ahorre dinero, y proporcionar líneas de unión de alta velocidad que vayan desde cada uno a la unidad central. Normalmente los microcomputadores individuales han sido construidos por un mismo fabricante para asegurar así su compatibilidad, aunque pueden encontrarse redes que tratan de unir microcomputadores distintos entre sí.

El tercer esquema, el anillo, utiliza hardware especial de interface que conecta cada computador con el anillo y, luego, una línea de unión de alta velocidad entre las interfaces: normalmente un cable coaxial. (El nombre “anillo” puede llevar a confusión, ya que el cable de conexión no tiene por qué cerrarse sobre sí mismo.) Hasta el momento, los anillos no han tenido una gran aceptación. Sin embargo, permiten, en principio, enlazar máquinas fabricadas por distintas compañías a una misma unidad central, lo que, a su vez, permite a las organizaciones que han adquirido éstas por partes, procedentes de distintas fuentes, unirlas todos entre sí para obtener una oficina electrónica.

Hasta el momento la principal dificultad con los anillos estriba (al menos me lo parece a mí) en que intentan ser demasiado inteligentes. Varios sistemas que se encuentran en el mercado pretenden que el usuario de un microcomputador pueda hacerse cargo de la pantalla de otro, utilizar la memoria sobrante en otra máquina para procesar sus programas y muchas otras cosas de este tipo, que resultan útiles y convenientes; cuando de lo que se trata es de unir main-frame y minicomputadores, en particular cuando hay un gran equipo de profesionales bien preparados para mantener toda esta delicada estructura. También proporciona muchas horas de diversión a los profesores en los laboratorios de informática, pero, en conjunto, resulta demasiado complicado para el simple usuario.

La tercera forma de enlazar los microcomputadores consiste en convertirlos en partes separadas de un gran computador. Esto significa que todos los procesadores comparten el trabajo y que toda la memoria es accesible a todos los procesadores. La idea se basa en que, por ejemplo,

si su procesador está parado, porque mientras usted está ejecutando un paquete de procesamiento de textos para escribir una carta, se detiene un instante, mirando al techo, preguntándose si llamará a su corresponsal “maldito mentiroso” o “víctima de una inexactitud terminológica”, y el procesador de su vecino tiene excesivo trabajo porque está ordenando un gran archivo, su procesador debería compartir el trabajo y servir de ayuda.

Veremos más adelante en las páginas 246-250 de qué modo la próxima generación de computadores podría utilizar muchos procesadores en paralelo, pero el soporte adecuado para esta clase de asunto es un chip, y no una oficina llena de gente ordinaria en su sano juicio.

LA OFICINA ELECTRÓNICA

En las dos últimas páginas hemos visto cómo los microcomputadores pueden unirse entre sí permitiendo a distintas personas compartir los mismos archivos. Esta tecnología tiene clara aplicación en las pequeñas empresas. Hay una enorme cantidad de pequeños negocios en el mundo y la mayoría de ellos hacen cosas estandarizadas que pueden automatizarse razonablemente. Consideremos uno de estos casos: una empresa de confección que diseña, fabrica y comercializa sombreros de alta costura para señoras. Tiene ocho departamentos y, para mayor comodidad, diremos que en cada uno hay un solo empleado:

Director; Secretaria; Vendedor; Diseñador; Jefe de almacén; Confeccionador; Tenedor de libros y contable

Los ocho tienen microcomputadores unidos a una unidad central provista de un disco. Además, todos ellos pueden acceder a un gestor de base de datos en el disco central que contiene los datos de la compañía. Algunos microcomputadores tienen su propia impresora u otro periférico, mientras que otros utilizan la impresora central en la unidad central. Para ver lo que ocurre, sigamos el trabajo diario a través de la red.

De lo que se trata en este ejercicio es de hacer dinero y el eslabón último en este loable empeño es el vendedor. Este telefonea a distintas personas, normalmente propietarios o compradores de las tiendas de modas, y las persuade de que compren los elegantes sombreros de la firma. Telefonea tanto a posibles clientes,

cuyos números de teléfono ha obtenido en algún lugar, como a otros tradicionales de la casa, y los persuade de que compren. Tiene un módem telefónico que le permite recoger los encargos directamente del computador de sus clientes.

Imaginémoslo llamando a un posible cliente. Le telefonea y trata de persuadirlo para que se interese por los productos de la firma. Si consigue un pedido, debe introducirlo en el sistema. Entrará el nombre del cliente, dirección, tipo de tiendas, etc., en un formulario que aparece en su pantalla. La información que entra se almacena en el disco y está a disposición de cualquier otro empleado.

Supongamos ahora que suena el teléfono. Es alguien a quien ya conoce y que quiere hacer otro pedido. Teclea su nombre en el formulario que tiene en la pantalla y escribe su pedido en el mismo. Esto crea un nuevo record en el disco, que tiene el nombre del cliente, el número único de identificación del mismo, para unirlo a un record aparte con su dirección y una descripción de su pedido: en este caso, media docena de cada uno de los grandes éxitos de la firma, el “Burbujas Punk” y el “Sueño de Tarzán”, el precio con el descuento que se hace al cliente, el lugar de envío y otros detalles relevantes.

Observemos a continuación a la jefe de almacén. Ésta tiene un programa que explora la base de datos en busca de records como el que acaba de crear el vendedor, que tiene un pedido pero ninguna indicación de que ha sido procesado. Cuando el programa encuentra un record de este tipo, comprueba en la lista de existencias si la firma tiene en almacén lo que se desea.

En nuestro caso, faltan tres “Sueños de Tarzán”, de manera que el programa crea un nuevo record para indicar el número de sombreros que deben confeccionarse para poder cumplir con el pedido. A continuación, la jefe de almacén entra los sombreros que tiene en su pantalla. El programa comprueba si hay alguna discrepancia entre la lista de stock y lo que tiene realmente. Luego hace varias cosas. Crea un record que indica que a la Sra. Riera se le han enviado 6 “Burbujas Punk” y 3 “Sueños de Tarzán”. Crea un record que dice que la Sra. Riera desearía conseguir 3 “Sueños de Tarzán” más cuando se tengan en stock. Imprime una orden de envío en la impresora que tiene a su lado, que incluye lo que se envía y una etiqueta con la dirección de la Sra. Riera. Ahora el empaquetador ya puede hacer el paquete. El programa añadirá en una lista los 3 “Sueños de Tarzán” que faltan para que se confeccionen en la fábrica.

Los sombreros, una vez confeccionados, se añaden a la lista de stock y se colocan en los estantes. Este programa también toma nota de los materiales utilizados para la confección del nuevo stock. Tanto el contable como otros programas que hacen los pedidos de más materiales, necesitan estos datos.

A partir de este momento, la base de datos tiene un record de que se ha enviado a la Sra. Riera algunas de las cosas que pidió. El tenedor de libros empieza su *show*. Su programa explora la base de datos en busca de records de pedidos y escribe una nota que indica que la Sra. Riera se ha convertido en deudora por la

cantidad de las mercancías que se le han enviado. El programa del tenedor de libros crea una factura para enviar a la Sra. Riera y toma nota de que se han enviado las mercancías.

Al mismo tiempo, llega el pago de Petit Pierre de París de unas baratijas que compró hace seis meses. El tenedor de libros lo entra en su máquina. El sistema encuentra el record en la base de datos que contiene la deuda y la elimina. Ya no existe ninguna razón para guardar un record de esta transacción particular, de modo que se borrará y el total aparecerá tan sólo en el libro mayor de ventas.

De vez en cuando el contable necesita saber lo que se ha hecho: lo que se ha comprado, lo que se debe, lo que se ha pagado. Su programa correrá como un loco por la base de datos averiguando todas estas cosas y presentándolas dispuestas de la manera que les gusta a los contables (y que normalmente nadie más es capaz de en tender).

El director también necesita ver parte de todo esto. Debe saber si el vendedor mantiene un movimiento de llamadas satisfactorio. ¿Se encuentra la compañía a menudo con falta de stocks (como ha ocurrido en el caso de la Sra. Riera) para cumplir con un pedido? Si es así, ¿por qué? ¿Es suficiente negocio vender estos exóticos gorros de piel, hechos por mujeres de las montañas nepalesas, como para justificar los fuertes gastos en viajes del diseñador a esta parte del mundo? ¿Cuánto tiempo tardan los clientes en pagar? ¿Se está convirtiendo la empresa en una especie de sociedad financiera en lugar de ser una empresa de confección de sombreros? ¿Cuántas deudas de importancia tienen? ¿Cómo mejoraron las ventas después de la campaña publicitaria del último mes? ¿Hizo algún efecto sobre las ventas la introducción a principios de año de la nueva línea de su competidor Miss Chou Chou? ¿Qué compra cada tipo de cliente? ¿Debería abrir una nueva línea de ventas en un mercado completamente distinto?

Para tomar estas decisiones, el director necesita mucha información, que la mayoría de las veces es mejor presentar en forma de gráficos e histogramas de tarta o de barras. Tiene a su disposición una gama de software que sacará información de la base de datos presentándola en alguna de estas formas. También dispone de paquetes de planificación financiera que descontarán las letras de cambio y realizarán perfiles de los deudores.

Su secretaria es más que un jefe de oficina. Confecciona la mayoría de los documentos que la compañía envía al exterior, tales como folletos o listas de precios, y para hacerlo utiliza una terminal de procesamiento de textos especial y una impresora de calidad. Sus funciones de secretaria han desaparecido casi totalmente, ya que se supone que cualquiera que quiera escribir una carta lo hará por sí mismo en su terminal, utilizando un paquete de procesamiento de textos.

El diseñador utiliza una máquina de gráficos independientes, con un panel digitalizador, unida al computador principal. La utiliza para poner a prueba las

nuevas formas en tres dimensiones y para especificar y cuantificar los materiales necesarios para las creaciones de la firma.

El computador está unido a cada uno de los terminales. Es una máquina de 16 bits, que hace funcionar un sistema operativo multiusuario y multitarea, y tiene dos discos de 40 MB. También tiene una impresora matricial de puntos de alta velocidad, para los documentos internos.

Por último, queda la función más importante: la copia física del disco. Una vez al día, o una vez cada semana, según cada usuario, debe copiarse el contenido de los dos discos en una cinta que se almacena en un lugar seguro, de modo que si se incendia la oficina, pueda disponerse todavía de los programas y archivos esenciales. Resulta mucho más fácil sustituir el hardware que los datos producidos por el software.

REDES DE LARGA DISTANCIA

La difusión de los computadores permitirá que estas máquinas se hagan cargo de muchas de las funciones que hoy en día realizan el teléfono, el télex y, en particular, el servicio postal. Se utilizarán para intercambiar facturas, cuentas, pedidos, memorándums, informes y todo tipo de conocimientos.

El servicio postal hace, lentamente y a mano, más o menos lo que desearíamos ver hecho electrónicamente. Imaginemos que confeccionamos un envío (una factura o un par de calcetines), lo empaquetamos bien y le ponemos la dirección que nos plazca. Sería perfecto si pudiéramos tratar los mensajes informatizados de un modo totalmente informal. Deberíamos ser capaces de encontrar una persona a partir de una dirección incompleta. Deberíamos encontrarla aunque estuviese de viaje por el país; en una red de datos adecuada la gente podría tener direcciones móviles. Sería magnífico que pudiésemos localizar al destinatario del mensaje no sólo a partir de la dirección de su domicilio sino también por sus negocios, asuntos públicos o intereses; bastaría, entonces, con que enviáramos un mensaje a todos los que tuvieran un computador Timex-Sinclair, o bien a todos los agricultores de una comarca determinada, o quizás a todos los miembros de un partido político. Los datos acerca de la dirección de una persona podrán decir mucho sobre ella.

Como dijo Gandhi refiriéndose a la civilización occidental: «Podría ser muy agradable», pero todavía no lo es.

El primer modelo que la gente considera para cualquier tipo de red electrónica es el sistema telefónico; simplemente porque hace tiempo que se conoce y ha tenido mucho éxito. En el sistema telefónico, un número determinado de abonados están conectados en un sistema en “estrella” a una central telefónica de conmutación. Las centrales de conmutación están a su vez conectadas en estrella a las centrales interurbanas, y las centrales interurbanas de distintos países están conectadas en estrella a las centrales telefónicas internacionales.

Es fácil ver cómo se encuentra un abonado particular: se busca, primero, el país, el área o central interurbana que le corresponde, su central telefónica local y el par de cables que llevan a la persona con la que deseamos comunicar. Tan sólo se necesita una docena, más o menos, de dígitos para especificar a cada uno de los varios miles de millones de personas conectadas al sistema telefónico internacional. El sistema tiene sus ventajas: concentra toda la “inteligencia” que necesita en las centrales donde la compañía telefónica puede mantenerla y protegerla. A la inversa, los aparatos telefónicos suministrados a los abonados pueden ser sencillos y, por tanto, baratos. En la época en que la inteligencia debía ser mecánica, este método era perfecto.

El sistema en estrella funciona como el tráfico rodado. Aunque se puede llamar a cualquier parte del mundo desde cualquier teléfono, el número de llamadas locales, es decir, a los abonados que están cerca, es mucho mayor. En un sistema telefónico la densidad de tráfico tiende a seguir la ley de Zipf (véase p. 123).

Sin embargo, el sistema en estrella también tiene sus desventajas. Utiliza una terrorífica cantidad de cables, puesto que cada abonado debe estar conectado, al menos, a la central.

Es difícil añadir un nuevo abonado porque debe instalarse nueva maquinaria en la central y dos cables que vayan hasta su casa. Aunque nos hemos referido a llamadas telefónicas que utilizan señales emitidas por la voz, podemos emplear la misma terminología para referirnos a los datos informáticos, ya que los datos pueden transformarse en señales sonoras y el so nido puede transformarse en datos (véanse pp. 178-181).

Desde hace muchos años, los computadores se han unido entre sí a través de canales de larga distancia. En fechas más recientes, se han conecta do en forma de red alrededor de un procesador único. Luego, los procesa dores se conectaron en redes.

Una de las redes más conocidas era la red ARPA *(Advanced Projects Research Aqency)* en Estados Unidos. Para entrar en el sistema se precisaba una conexión para el procesador adecuado, el conocimiento de los protocolos y el proceso era, en general, complicado. También era un proceso muy caro y normalmente sólo lo podían utilizar gente que trabajaba en las universidades o los militares.

En estas redes, los datos se recogen normalmente en paquetes de longitud estandarizada, que se envían por la línea uno tras otro como los vagones de un ferrocarril. Todo el proceso resulta en conjunto muy poco ágil. Las oficinas de correos de los países industrializados tienen en su mayoría planes para ofrecer autopistas de datos a través de sus propios computadores.

Sin embargo, no resulta fácil el acceso a estos sistemas, que, además, son caros y no tienen la flexibilidad del servicio de correos. A nivel popular se empezó (especialmente en Estados Unidos) a improvisar redes utilizando el sistema telefónico.

Como vimos con anterioridad, la transmisión de datos por teléfono es lenta y poco fiable. Por otra parte, es muy rígida. Puede flexibilizarse elaborando un software que dirija el aparato telefónico, es decir, marque el número, escuche los distintos tonos y compruebe que se ha llamado al computador (o persona) adecuado. Por ejemplo, un hombre que instale un sistema automático para controlar cincuenta estaciones de televisión distantes, sin personal humano hará que su computador llame a cada uno de ellos durante la noche e indique qué canales de televisión debe emitir la estación durante el día siguiente y a qué horas.

En Gran Bretaña, el British Telecom’s Prestel proporcionaba una red de datos accesible a cualquier persona conectada al sistema que se tomase la molestia de conectar con ella. Pero las pantallas son de tan baja calidad y tan difíciles de encontrar que ha tenido poca aceptación. Entonces, el British Telecom’s ofreció un servicio de “puerta” que unía por teléfono cualquier computador con otro, a través del de la propia compañía. Pero esto continuaba siendo demasiado caro y funcionaba con dificultad. Hasta ahora aún no se han empezado a utilizar todas las posibilidades que ofrecen los computadores para la comunicación a gran escala.

Desde un punto de vista técnico, sería perfecto que cada uno de los millones de computadores que hay en el mundo pudiera transmitir datos a cualquier otro. Pero esto resulta difícil de creer, por lo que considere-

mos en primer lugar cómo podríamos lograr que un pequeño número de máquinas se comuniquen entre sí. Esencialmente, existen dos modos.

El primero es la conmutación en la que se reproduce el sistema en estrella de la red telefónica, de manera que cada terminal conecta a una central de conmutación, cada central de conmutación a otra de nivel más alto y así sucesivamente. El segundo modo de transmitir los datos es mediante buses (véase p. 28). Todos los usuarios están conectados a un flujo de datos y pueden buscar cualquier mensaje que vaya dirigido a ellos.

A medio plazo el primer sistema deja mucho que desear. Es poco flexible y tiene muchas limitaciones; además, las centrales son muy complicadas. Por otro lado, cabe preguntarse ¿por qué construir centrales para conmutar los datos cuando el propio microcomputador individual ya posee la inteligencia necesaria para hacerlo?

El segundo método exige, por el contrario, una infraestructura técnica menos compleja. Lo único que hay que hacer es presentar todas las señales del sistema a cada terminal. Las señales les llegan de varias formas distintas. En uno de los sistemas, una unidad central controla la red, que podría consistir en un cable único de fibra óptica.

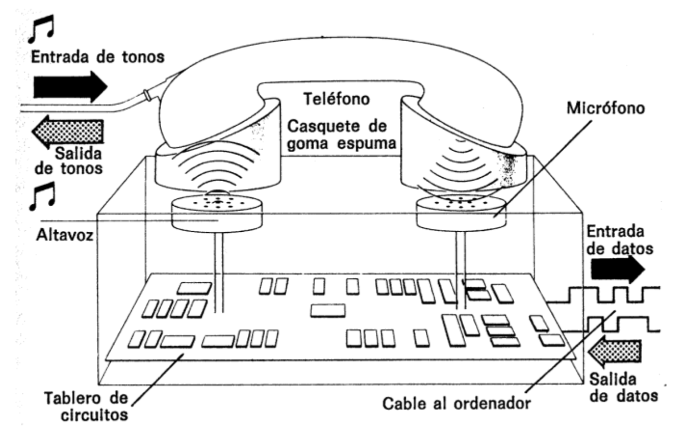


Fig. 29. Un módem toma entradas del computador en forma de bits, transforma los 1 y 0 en tonos de distintas frecuencias y los envía a la línea telefónica. En un módem de aficionado, sencillo, como éste, el transmisor y el receptor del teléfono se colocan en dos pequeños cubiletes de goma. Los módem más profesionales prescinden de los altavoces y se conectan directamente a la línea telefónica.

La red conecta físicamente a todo el mundo entre sí. Es como si todos utilizasen radios: cualquier cosa que uno de los usuarios transmita puede ser captada por los demás. No se necesitan, al menos en el sistema más simple, centros de conmutación como los de las centrales telefónicas. Todo esto lo realizan terminales individuales. Si consideramos, por ejemplo, una red que cubra todas las islas británicas, nadie estaría a más de 1.500 km de distancia de otra persona por fibra óptica, lo que significa que el tiempo que tardaría una señal para ir a la parte más alejada de la red y volver sería de media milésima de segundo.

Si nadie tiene nada que decir no hay transmisiones. Supóngase que el terminal A quiere enviar un mensaje al terminal B. Cada uno tiene un número de identificación único, como si fuera un número de teléfono. En primer lugar A tiene que conseguir que la unidad central lo atienda. Transmite su número seguido del número de B. La unidad central realiza una suma de comprobación con los números (véanse pp. 13-15) para asegurarse de que los números son válidos (no han sufrido ninguna perturbación) y retransmite la llamada. A escucha para ver si la llamada ha sido retransmitida adecuadamente. Todos escuchan a la unidad central durante todo el tiempo. Puesto que esta llamada es para B, los otros no intervienen. Si el terminal B ha sido activado y funciona bien, transmite su número como acuse de recibo. La unidad central comprueba que no se ha producido ninguna perturbación (que podría haber tenido lugar por un defecto en el equipo B) y dice a A que empiece. Entonces A transmite su mensaje a B. Cuando termina, envía a continuación una señal de “libre”, que la unidad central retransmite a toda la red para indicarles que ya pueden enviar su señal de «Tengo un mensaje para fulano de tal» si así lo desean.

El lector perspicaz se preguntará qué ocurre si A y, por ejemplo, C envían la señal «Quiero enviar un mensaje» al mismo tiempo. Hay una colisión o choque. La unidad central recibe las dos señales sobreimpresas y al tratar de hacer su suma de comprobación, fracasa y se bloquea. Ni A ni C oyen su señal retransmitida, por lo que ambos lo intentan de nuevo. Para evitar otra colisión, cada uno de ellos espera un plazo de tiempo aleatorio (calculado estadísticamente a partir de la densidad de tráfico) y retransmite la señal.

Como el intervalo escogido por A y C en el caso anterior ha sido elegido al azar, lo más probable es que no vuelvan a retransmitir al mismo

tiempo. Cualquiera de las dos que llegue en primer lugar logra que el otro espere hasta que haya terminado.

El primero en desarrollar este sistema fue la red Aloha (“hola”, en polinesio) de computadores, en Hawái, a mediados de los setenta.

Aloha utilizó cable en lugar de fibra de vidrio, pero el principio continúa siendo el mismo. Podría pensarse que el sistema pasa la mayor parte del tiempo evitando colisiones. Sin embargo, resulta que si la capacidad de manipulación de datos es lo bastante grande, las colisiones no son un verdadero problema, ya que las señales «Tengo un mensaje» son tan cortas que la probabilidad de que una colisione con otra es pequeña. La red puede manejar cerca del 80% de su capacidad teórica, que correspondería, evidentemente, a la que tendría si todos emitiesen exactamente uno detrás de otro sin ninguna interrupción.

La alternativa consistiría en sondear (véanse pp. 38-40) cada terminal por turnos para comprobar si tiene un mensaje. Así se despilfarra tiempo en los terminales inactivos y se deja sin la suficiente atención a los más activos. Además, la unidad central debe saber quién está en la red y quién no. En el esquema Aloha podemos unirnos a la red y dejarla a voluntad.

Este sistema tiene una enorme capacidad de procesamiento de datos. En la actualidad las fibras ópticas pueden transmitir hasta 300 Mb/s. Tomando un factor de carga realista del 80%, tenemos 240 Mb/s o 30 MB/s. Una forma de considerar esta capacidad es mirando la parte más lenta de la interface de información: el ojo humano en la lectura, el dedo humano en la escritura. Más pronto o más tarde todo lo que hay en la red deberá pasar por estos dos canales. Pocos pueden leer y absorber más de, por ejemplo, 5.000 palabras en un día. Pocos escritores profesionales producen más de 1.000 palabras de texto terminado por día, pero también tenemos que considerar los documentos comerciales y los folletos de propaganda. Como promedio, 5.000 palabras de texto por día sería una generosa asignación para cada uno de los usuarios de la red. Así tenemos como datos de trabajo unas 5.000 palabras que equivalen a 30.000 caracteres en inglés y 40.000 en castellano (cada palabra en inglés tiene de media 6 caracteres y en castellano 8) por usuario y día de funcionamiento. Esto da como promedio en el caso inglés 1/3 B/s, de modo que una fibra óptica única que pasase por toda Gran Bretaña podría proporcionar los datos que necesitan cerca de 90 millones de personas.

En la práctica, ocurre en muy pocas ocasiones que alguien desde Londres, por ejemplo, quiera enviar un mensaje a Orkrey (véase Ley de Zipf, p. 123), de modo que resultaría probablemente mucho más apropiado tener redes confiadas a áreas geográficas determinadas, con centros de conmutación que se remitieran los mensajes entre sí. En largas distancias, las redes de datos utilizarán probablemente fibra de vidrio bajo los océanos, satélite o microondas de radio indistintamente. Estos canales de alta capacidad resultarán muy caros, por lo que deberán de operar de forma más sistemática. Los datos tendrán que ser recogidos, compactados y enviados en bloques, tal como se hace en la actualidad en los sistemas de grandes computadores.

Sin embargo, lo atractivo del sistema Aloha está en que las direcciones no han de ser geográficas. Cuando marcamos un número de teléfono, en realidad llamamos a una casa o a una oficina, a algo inanimado que no puede hablarnos. Naturalmente, cuando llamamos esperamos que la persona con quien deseamos hablar esté realmente allí, pero a veces está y a veces no.

Lo que queremos es poder llamar a una persona sin necesidad de saber dónde está. En el sistema en estrella esto resulta muy difícil; todo el mundo debería dejar antes las direcciones que indicasen sus cambios de lugar en cada instante. Pero debido al ritmo de la vida actual, podemos decir que iremos a casa de Luisa al mediodía y encontrarnos con que hemos tenido que ir a la de Juan, con lo que hubiéramos tenido que dejar otro número en el número de Luisa. Al final el sistema se habría vuelto tan complicado que se nos escaparía de las manos, en particular si intentásemos trasladarnos desde una central en un sistema en estrella a otra.

En el sistema Aloha, donde toda la inteligencia está en los terminales, el “direccionamiento” resulta muy fácil. Un terminal podría tener un número fijo correspondiente a su dirección, pero también podría tener números que correspondiesen a la gente que tiene más cerca. Así, cuando llegáramos a casa de Luisa, programaríamos su terminal para recibir nuestras llamadas. Si nos trasladamos a casa de Juan haríamos lo mismo. La llamada telefónica de la Srta. Fernández o su texto-mensaje de computador nos llegaría volando sin ningún problema. Sus facturas informatizadas encontrarían a sus deudores estuviesen donde estuviesen.

El sistema de direcciones podría ser aún más general, de manera que si formásemos parte de grupos de intereses (tanto recreativos como pro-

fesionales) podríamos obtener, por ejemplo, todos los mensajes para los miembros de nuestra sociedad. Podríamos suscribirnos a revistas, periódicos, diarios o a simples informaciones distribuidas de forma digital a través de nuestro computador. Los miembros del grupo podrían también seguirnos por todas partes si así lo indicáramos (tan sólo tendríamos que decir a la máquina dónde estamos o estaremos para comunicarnos con ellos). Sin duda, las posibilidades son asombrosas.

Fibra óptica

Durante la década actual asistiremos a una rápida transición del cable de cobre y las conexiones por radio, tanto directas como vía satélite, a la fibra óptica. Hoy día se están tendiendo miles de kilómetros de fibra óptica en Estados Unidos y en Europa, y el primer cable submarino de fibra óptica se espera que esté terminado en 1988, conectando Japón con Hawái. Una fibra óptica consiste en un minúsculo hilo de vidrio muy puro a través del cual puede transmitirse una señal luminosa. La luz se genera mediante un láser o un diodo láser y se pulsa digitalmente para poner en código la información, que puede ser datos de computador, sonido o emisión televisiva.

El vidrio tiene tanta pureza que la señal puede transmitirse hasta 35 km (o 20 millas) sin reamplificarla (los cables corrientes de cobre necesitan un amplificador cada kilómetro o incluso menos). La capacidad de datos de una fibra está limitada por los semiconductores utilizados para generar y detectar los impulsos luminosos, y es de unos 300 Mb/s más o menos.

A medida que mejora la tecnología de los semiconductores la capacidad de datos de las conexiones existentes podrá aumentar fácilmente. Y, a medida que el vidrio reemplace al cobre, el metal rojo existente en el mundo se irá recuperando del subsuelo de nuestras ciudades.

BASES DE DATOS ENORMES

Es muy probable que la difusión de los computadores en oficinas y hogares y la instalación de redes de datos de alta velocidad por los distintos servicios de telecomunicaciones nacionales, tengan un efecto dramá-

tico en el almacenaje y recuperación de la información. Hoy en día, para encontrar algún dato que no conocemos, debemos buscar en una biblioteca. Podríamos ir a la biblioteca de nuestro barrio y buscar en el *Diccionario de la Real Academia Española* o en el *Salvat Universal,* por ejemplo. Si no fuera suficiente, buscaríamos en otro diccionario o enciclopedia y luego empezaríamos a mirar los estantes de libros. Si se quiere un tipo de información que cambia rápidamente, como los horarios de los trenes, consultaríamos un actualizado^ un anuario o un suplemento, que se publican con mucha más periodicidad que una enciclopedia. Si queremos saber cómo anunciar comida para animales en Filipinas, o encontrar los proveedores capaces de suministrar tuberías de fundición en Bolivia, deberíamos consultar una guía comercial.

Todos estos datos son informaciones reunidas por alguien y publicadas; todo cuanto debe hacerse es encontrar la publicación adecuada. Pero existen otras cosas que se necesita saber urgentemente y que podrían muy bien no haber sido recogidas ni publicadas; por ejemplo, el precio medio de las acciones de las compañías de Hong Kong que realizan intercambios comerciales con China, o las ganancias medias (basadas en las declaraciones fiscales) de las compañías informáticas y de las empresas de componentes electrónicos de la República Federal de Alemania.

Estas posibilidades técnicas revolucionarán inevitablemente los stocks mundiales de información. Habrá, como ya ocurre actualmente, una tendencia hacia la publicación electrónica de información “caliente”, tal como los precios de las acciones, resultados deportivos, carteleras de espectáculos, horarios de aviones, ferrocarriles y autobuses, etc.

Este proceso no se detendrá. La gente ya empieza a ver que información tradicionalmente publicada en papel, como los indicadores y las previsiones económicas, podría difundirse en forma de datos para computadores personales, y ejecutarse con un gestor de base de datos. Por ejemplo, un boletín informativo sobre las economías africanas sería mucho más útil para sus suscriptores en forma de base de datos que se ejecutase en sus microcomputadores, de manera que pudiesen hacer preguntas como «¿Quién es el ministro de Finanzas de Kenia?» o «¿Cuál es el valor de las exportaciones de pescado de Gambia?» Las nuevas ediciones del boletín podrían enviarse en forma de discos flexibles o de datos por teléfono, para actualizar la base de datos de los usuarios.

Antes de que esto ocurra deberá pasarse a forma digital el stock mundial de información impresa, lo que sin duda es una tarea harto compleja.

En una segunda etapa, probablemente este proceso se automatizará y se dispondrá de instrumentos inteligentes de software que hagan el trabajo de los auxiliares de investigación. Cuando pueda disponerse de todas las fuentes importantes de información del mundo, se dirá al software auxiliar que se ejecuta en el propio computador: «Ve y busca todo lo que puedas encontrar sobre el pintor Turner»; o «Prepara un informe sobre el mercado mundial del yute». El pobre software deberá consultar probablemente una gran serie de bases de datos secundarias, que catalogarán las fuentes de datos principales. Entonces las llamará y les enviará mensajes con sus peticiones: «Dime esto, ¿dónde puedo encontrar lo otro?» Una vez obtenidas las respuestas, las verificará y clasificará para que se puedan utilizar. El avispado software en cuestión puede perfectamente no imprimir un informe completo, sino presentar de nuevo los resultados como minibases de datos a las que puede acudir y consultar la persona que ha efectuado la pregunta.

Este software será muy avanzado si lo comparamos con los gestores de datos actuales. Deberá estar dotado de mucha inteligencia para lograr una visión general del tema y encontrar las mejores estrategias para averiguar más detalles. Deberá ser capaz de moverse por el mundo de la información de forma parecida a como lo hace un perro que coge los objetos que le lanza su amo y se los trae de vuelta.

También habrá una enorme necesidad de equipos automáticos de lectura, ya que el stock de información en papel existente en el mundo deberá transcribirse a forma electrónica. La Biblioteca Británica tiene alrededor de 13 millones de libros, con una media probable de 60.000 palabras o 360.000 caracteres por libro. En el mundo deben existir otros veinte centros como éste, lo que totaliza unos 93 billones de caracteres que deben transcribirse. Si a éstos añadimos los 5.000 periódicos que se editan diariamente en el mundo, que contienen unas 100.000 palabras o 600.000 caracteres cada uno, y multiplicamos esta cifra por tres para incluir el número de revistas técnicas y científicas que se publican diariamente, y si consideramos además que se ha de transcribir lo acumulado en los últimos cien años, obtendríamos la cantidad de bytes que deberán pasarse a código ASCII, para que el stock mundial de conocimientos esté completamente automatizado. Este trabajo deberá ser efectuado por

máquinas. Sin embargo, estas máquinas todavía no existen, ni en cantidad suficiente ni con la potencia necesaria para hacerlo. Por otra parte, también necesitaremos software que pueda traducir de una lengua a otra.

Pero esta espléndida perspectiva (tener a nuestra disposición los conocimientos acumulados durante toda la historia de la humanidad con sólo pulsar un botón) presenta una dificultad. Como el conocimiento es poder, hay gente interesada en impedir su libre acceso. Ejemplo de ello es el archivo estatal de España, que incluye registros de los galeones hundidos cargados de tesoros. Mientras nadie sabía nada sobre ellos, se permitía el acceso a los investigadores. Muy pocos sabían cómo trabajar en el archivo o cómo leer los documentos antiguos escritos a mano. Ahora que se ha desvelado el secreto, el archivo es la «Meca» de los buscadores de tesoros. Muchos otros filones de información valiosa yacen intocados porque son demasiado difíciles de investigar y nadie sabe que existen. Cuando los investigadores automatizados puedan empezar a “excavar” en cualquier lugar, aparecerán, casi con toda seguridad, grandes presiones para restringir y controlar su acceso.

La base de datos se traducirá en un código secreto y su clave será conocida tan sólo por un limitado número de personas autorizadas. Esta situación será una verdadera lástima y significará el final de una tradición académica que cuenta con muchos siglos de existencia.

Codificación

Sin duda la necesidad del secreto y de códigos en clave es un fenómeno que ya encontramos hoy día. El problema con los datos electrónicos reside en que los transmisores pueden copiarlos fácilmente. El papel tiene sus virtudes; una de ellas es la durabilidad y unicidad de los documentos. No existe ningún equivalente electrónico a un billete de banco que pueda pasar de un computador a otro sin ser copiado. En su lugar debemos tener un esquema que asegure que los mensajes sólo pueden pasar por dos personas y que nadie puede interferidos. Un banco puede decir a otro que pague una cantidad de dinero a alguien. Ambos guardan un registro cronológico de los mensajes y puede verificarse si ha ocurrido todo como estaba previsto.

Dada la potencia de los computadores, no hay ningún problema en mezclar textos y números tan íntimamente que las mayores máquinas tardarían mucho tiempo en separarlos de nuevo con procedimientos aleatorios. Sin embargo, existe una alternativa mejor que proviene de los desarrollos realizados en el mundo de las matemáticas en los últimos diez años. Consiste en un tipo de codificación llamado *Trap-door coding,* que se basa en procedimientos matemáticos que son fáciles de realizar en un sentido pero muy difíciles en el inverso. Uno de estos casos sería, por ejemplo, encontrar los factores primos de un número grande (200-300 dígitos). Para hacerlo se necesita normalmente una enorme cantidad de tiempo; en cambio, si se conocen los factores, es un juego de niños multiplicarlos entre sí para obtener el número. Este principio puede utilizarse para escribir un sistema de codificación en el que, paradójicamente, puede publicarse el código, pero en el que sólo la persona que lo ha ideado puede leer los mensajes escritos con él. Si se enlazan dos códigos de este tipo, las partes en correspondencia (por ejemplo, un banco y sus clientes) pueden estar seguras de que el mensaje recibido procede forzosamente de su corresponsal.

4. Progresos

UNA REVOLUCIÓN EN EL PENSAMIENTO

Una de las razones por la que resulta difícil familiarizarse con los computadores es que la informática incorpora algunas perspectivas que suponen un cambio radical respecto a la visión consensual del mundo que hemos heredado de los temerarios y confiados científicos del siglo pasado. Estos cambios se han incorporado a las tradiciones de la informática. Las personas que están dentro de este sector los han asimilado inconscientemente; quienes están fuera se sienten abrumados sin saber exactamente por qué. El esfuerzo intelectual necesario para informatizarse recuerda las convulsiones que revolucionaron las matemáticas en la primera mitad de este siglo.

La causa inmediata de esta revolución consistió en tres preguntas que David Hilbert planteó a la comunidad matemática en 1900. A saber: ¿Constituyen las matemáticas un sistema completo en el sentido de que cualquier afirmación formulable dentro de ellas puede demostrarse cierta o falsa? ¿Son las matemáticas consistentes en el sentido de que no puede llegarse nunca a una afirmación «falsa» deduciéndola mediante una serie de pasos válidos? ¿Son las matemáticas decidióles, es decir, existe un método aplicable, por lo menos en principio, a cualquier afirmación para decidir si es verdadera o falsa?

Un joven matemático húngaro llamado Kurt Gödel respondió en 1929 a las dos primeras preguntas. Para formular su respuesta, inventó lo que en su momento a mucha gente pareció una forma delirante de codificar reglas matemáticas y fórmulas como números. Dio a cada afirmación matemática un número y después demostró que cualquiera que fuese el sistema de reglas que se adoptase para las matemáticas, siempre habría números de código extras (es decir, afirmaciones extras) que no podían derivar de las ya existentes. En otras palabras, en cualquier sistema lógico (no únicamente en matemáticas) siempre existirán afirmaciones cuya validez o falsedad no puede demostrarse en función de las afirmaciones anteriormente establecidas. También mostró que es imposible demostrar

la consistencia de ninguna parte de las matemáticas sin introducir reglas nuevas desde fuera.

Estos descubrimientos apenaron a los matemáticos, pero favorecieron a la ciencia informática que estaba a punto de nacer, ya que en el proceso de su demostración Gödel abrió una brecha en el muro que parecía separar las reglas y fórmulas matemáticas de los números que generaban. Simplemente dijo: «Toda regla es también un número.» En su momento esto resultaba tan extraño que parecía falso; hoy en día nos parece trivial. Cuando se escribe la línea en MBASIC:

```
100 IF A > 34 THEN C=COS(K)ELSE C=SIN(K)
```

todo lo que ve el computador es el número hexadecimal siguiente:

```
FF D 60 64 0 8B 20 41 EF F 22 20 CF 20 43 FO FF 8C 28 4B 29 20 3A
    A2 20 43 FO FF 89 28 4B 20 0 0 0
```

A cualquier fórmula que pueda escribirse corresponde un número hexadecimal distinto, y no hay nada que impida sumar esos números o disponerlos ordenadamente.

Esta desenvuelta actitud en relación con los símbolos fue uno de los cambios necesarios antes de que pudiera desarrollarse la informática.

La segunda condición a priori necesaria fue desembarazarse de la idea de la «máquina de calcular». Casi tan pronto como el hombre aprendió a contar (y esto supuso una conquista intelectual tan grande como todas las que le han seguido), empezaron sus intentos de automatizar los procesos aritméticos. A medida que los ingenieros adquirieron más y más habilidad, las máquinas de calcular se hicieron más complicadas. Pero, al igual que los científicos, fueron cada vez mejores pero más escasas. Cualquier cambio en la forma de trabajar de una máquina exigía un laborioso proceso de reconstrucción.

También este paso se dio a partir de los desafíos de Hilbert. Alan Turing, un tímido y desgarbado joven matemático del King's College de Cambridge, se enfrentó al tercer problema a mediados de los años treinta. Gödel había abierto una brecha en el muro, pero fue Turing quien lo derribó. Argumentó que si fuese posible disponer de un método para probar cualquier teorema, todo lo que habría que hacer era poner a un matemático (un «calculador», como se le llamaría en este caso) a aplicar

las reglas. Para ver qué ocurriría a continuación, Turing eliminó el matemático humano y lo sustituyó por una máquina imaginaria para que hiciese el trabajo. Quizá la máquina fuese más estúpida que cualquier ser humano, pero al final hubiese llegado al mismo resultado.

La máquina tomaría cualquier teorema matemático (una serie de símbolos) y trabajaría a partir de ellos para determinar de forma totalmente mecánica si el teorema era o no demostrable. En otras palabras, debía trabajar como un computador, aunque en la época todavía no existían las máquinas que hoy conocemos con este nombre.

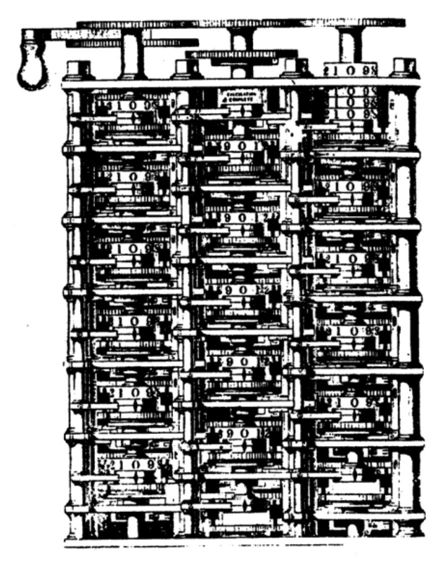


Fig. 30. Babbage con su *Máquina de Diferencias* se proponía construir las tablas de logaritmos simplificando las complejas ecuaciones algebraicas que se debían realizar para su obtención.

Quizá sin conocer la trascendencia de su innovación, Turing diseñó su máquina sin los engranajes, palancas y ejes que los diseñadores de máquinas de calcular anteriores a él se habían visto obligados a utilizar. Su dispositivo ideal era tan simple que rayaba en la ingenuidad. Todo el ingenio estaba incorporado a lo que hoy en día llamaríamos el software.

La máquina de Turing consiste en una cinta de papel infinitamente larga dividida en cuadros. La máquina tiene un cabezal mecánico que puede moverse indefinidamente cuadro a cuadro en cualquier dirección. Puede leer, borrar y escribir símbolos en los cuadros sobre los que se encuentra. De acuerdo con el símbolo que lee, puede cambiar su «esta-

do»; hoy día diríamos que puede determinar uno de sus flags de registro internos. Según su estado, reaccionará de distinta manera al leer el próximo símbolo. Esta capacidad de un programa para examinar determinados datos y después actuar en función del resultado constituye una idea esencial que Turing obtuvo de la «Máquina analítica» de Babbage.

Una vez se dispone de un «programa» para determinar la probabilidad de cualquier teorema matemático, la máquina se moverá adelante y atrás, solucionando el problema. Si el teorema es demostrable la máquina se parará; en cambio, si no es demostrable, no.

Turing prosiguió utilizando su máquina conceptual (conjuntamente con un resultado previamente obtenido por Cantor) para demostrar que no puede existir ningún método que permita resolver todos los problemas matemáticos.

Los matemáticos podían, en cierto sentido, alegrarse de este resultado, porque les permitía pensar que siempre habría trabajo para ellos. Pero su importancia real (que por supuesto no se apreció en su momento) consistió en establecer las bases de los computadores actuales.

La tira de papel es la memoria unidimensional de hoy en día que puede contener a la vez datos y programa. El cabezal que se desliza a lo largo de ella es el procesador. La única diferencia (y no se trata de una diferencia fundamental) está en que el procesador puede acceder directamente a cualquier posición de memoria, sin tener que avanzar paso a paso a lo largo de la cinta. Toda la inteligencia de la máquina está en las instrucciones contenidas en la cinta.

Los lógicos del siglo XIX habían demostrado que todos los procesos matemáticos podían construirse a partir de pasos lógicos simples. La segunda contribución importante de Turing (que ahora nos parece trivial) fue demostrar que una máquina capaz de realizar operaciones lógicas sencillas podía hacer cualquier cosa en matemáticas y por tanto imitar a cualquier otra máquina. Con materiales tan simples y aparentemente poco prometedores, construyó mentalmente una máquina universal.

La informática disponía ahora de una base sobre la que desarrollarse. Inicialmente, este desarrollo se apoyó en dos pilares muy simples. El primero era una gran losa sobre la que podía leerse el eslogan: «Todo lo que existe en el Universo puede reducirse a una serie de caracteres carentes de significado». El segundo pilar era otra losa en la que se leía: «Todo proceso del Universo puede reducirse a una serie de caracteres carentes

de significado». Cualquiera que se haya familiarizado con los computadores ha asimilado, consciente o inconscientemente estas dos ideas. Sin embargo, es comprensible que las demás personas difícilmente puedan sentir simpatía por estos puntos de vista.

A. Turing trabajó en la Universidad de Princeton con J. von Neumann y allí conoció a Claude Shannon, cuyo libro *La teoría matemática de la comunicación* es uno de los clásicos del siglo XX. Shannon introdujo el bit, unidad elemental de información que constituye la respuesta a una pregunta de Sí o No. Esto representó un paso esencial en el desarrollo de la informatización.

En 1937, un ingeniero alemán llamado Konrad Zuse construyó la primera máquina de calcular binaria. En ese mismo año, Turing, trabajando independientemente, utilizó la misma idea para construir una máquina de multiplicar que utilizaba relés electromecánicos. Entonces estalló la guerra y, mientras Zuse y sus calculadoras eran marginados por los proyectos de desarrollo de los cohetes V, Turing se incorporaba al equipo de matemáticos que trabajaron, en los primeros años de la segunda Guerra Mundial, en un proyecto anglo-norteamericano altamente secreto para descifrar mensajes en clave.

En 1939 el trabajo de descifrar claves se efectuaba sin más ayuda que el lápiz y el papel. Pero el volumen de material que debía examinarse en el centro británico de Bletchley Park hizo imprescindible la automatización. Turing y sus colegas pronto construyeron máquinas electromagnéticas para descifrar señales alemanas y, a mediados de la guerra, disponían de un dispositivo que realmente respondía a la orden mágica de comprobar y actuar en consecuencia, el primer rasgo de un auténtico computador.

En Estados Unidos se construyeron máquinas similares para descifrar códigos y para calcular la trayectoria de proyectiles de artillería (véanse pp. 126-129).

Posteriormente, Turing abandonó el trabajo de descifrar códigos y construyó una máquina completamente electrónica para perturbar emisiones radiofónicas, que constituyó probablemente la primera máquina útil que empleaba métodos binarios.

Inmediatamente después de la guerra, se construyó en Estados Unidos una máquina, ENIAC, que utilizaba 18.000 válvulas electrónicas para almacenar 20 números. Se desencadenó una carrera para intentar el desa-

rrollo de esta tecnología, pero los competidores fueron sorprendentemente lentos. La guerra redujo a escombros gran parte de la civilización occidental y los "vencedores" tenían problemas demasiado urgentes para ocuparse de los sueños de matemáticos excéntricos. Los ingleses construyeron lentamente en Manchester una máquina aislada y embrollada, cuya gran innovación consistió en utilizar pantallas de rayos catódicos para el almacenamiento de datos (véanse pp. 40-43) Para almacenar un bit, el cañón de electrones "escribía" un punto de carga en determinada posición sobre la pantalla. La carga permanecía en la posición durante determinado tiempo y podía ser "refrescada" antes de su definitiva desaparición. En Estados Unidos se intentaron resolver los mismos problemas, aunque los resultados que se obtuvieron fueron menos satisfactorios. Los expertos a ambos lados del Atlántico pensaban que en el mundo podría haber trabajo para cinco o seis de estas máquinas.

La historia del desarrollo de los computadores ha sido desde entonces mucho menos clara. Son pocas las ideas revolucionarias que se han introducido: el progreso se ha producido más bien a través del perfeccionamiento ininterrumpido de los detalles, que ha conducido a la obtención de más y mejores máquinas de coste mucho más reducido. Cada vez más es el propio usuario el que diseña la máquina que utiliza. Para explicar esto tendremos que retroceder de nuevo en el tiempo, pero a un período no muy lejano.

Tendencias actuales

El presente, incluso en nuestra época, sólo adquiere su sentido si lo con templamos a la luz del pasado. Lo mismo ocurre con el desarrollo de los computadores y de la informática, que ha ocupado a muchos de los mejores talentos de los últimos cuarenta años en una tarea en extremo interesante. Como cualquier otro campo en rápido desarrollo, ha madurado de forma un tanto aleatoria. Se ha intentado planificar este desarrollo, pero la experiencia demuestra que cuando se había conseguido un acuerdo acerca del plan, y éste había sido publicado, el desarrollo ya había seguido otro camino. Sin embargo, de este crecimiento vigoroso han surgido muchos conceptos básicos, muchos dispositivos, interconexiones

y lenguajes. Quien desee moverse con seguridad en el mundo de los computadores necesita tener algunos conocimientos de todas estas cosas.

Si no se hubiese inventado el transistor, probablemente los computadores serían todavía extrañas curiosidades. Sin embargo, los transistores, que son mucho más pequeños, mucho más fiables y se calientan mucho menos que las lámparas de radio, permitieron que los computadores empezasen a reducir su tamaño y su precio y a aumentar en potencia, de manera que hoy día es posible disponer de un computador personal de considerable capacidad por el precio de una máquina de escribir mecánica.

Al mismo tiempo que se abarataban y reducían de tamaño los transistores, lo mismo ocurría con los dispositivos de almacenamiento que los computadores usan como archivos. Mientras se escribía este libro, aparecían en el mercado las primeras unidades de arrastre de discos duros, de tamaño muy reducido, que proporcionan (a precios razonables y con la posibilidad de colocarlas sobre la mesa de trabajo) a cada máquina una capacidad de almacenamiento de 2 o 3 millones de palabras. Dentro de pocos años, los discos de láser darán a cada usuario una capacidad de almacenamiento de varios millones de palabras por el mismo precio que los discos actuales.

En la fabricación de chips el proceso de abaratamiento y miniaturización ha continuado sin interrupción durante los últimos veinte años. Como los computadores no son más que grandes cantidades de transistores reunidos en chips que están adecuadamente conectados entre sí sobre tableros de circuitos, este proceso ha comportado que los computadores sean ahora más pequeños y baratos, o más potentes por el mismo precio. Esta tendencia ha seguido una evolución tan regular que se ha convertido en una especie de ley de la naturaleza el hecho de que cada diez años se obtienen cien veces más transistores en un chip. Esto significa que cada década, por el mismo precio, los computadores multiplican su potencia por un millón. Lo que al principio parecía una simple y marginal tendencia en el desarrollo tecnológico, ha producido resultados inesperados y sorprendentes para la humanidad. Por ejemplo, considérese la variedad de computadores personales que existe en el mercado, Babbage, Gödel y Von Neumann se quedarían estupefactos.

Cada máquina tiene su propio BASIC. Toda persona que escribe un Interpretador de BASIC piensa que puede mejorar el esquema original.

En cada máquina la pantalla se maneja de forma distinta y muchas máquinas tienen unidades de discos con características particulares. Podría pensarse que el diseño del teclado no permite grandes excentricidades, pero no es así; es muy difícil predecir qué teclas contendrá un teclado que no se conoce. Incluso algo tan simple como la tecla DELETE (borrar) varía de una máquina a otra; escribir una rutina que borre siempre el carácter a la izquierda del cursor no tiene nada de sencillo.

Cualquiera que haya pasado más de media hora con un computador sabe que, cuando se ha escrito algo, debe pulsarse la tecla RETURN (retorno) para que tenga efecto. ¿No es así? En muchas máquinas la tecla de retorno está indicada como *NEWLINE o ENTER.* En algunas máquinas existen dos, una con cada nombre. A un nivel más profundo, puede suponerse que cada tecla envía un solo carácter a la máquina. Esto casi siempre es cierto; sin embargo, RETURN, NEWLINE y ENTER envían uno que en realidad se interpreta como dos: un avance de línea (ASCII 10) y una orden de retorno de carro o “ir al margen de la izquierda” (ASCII 13). Esto puede producir gran confusión si se está escribiendo un programa que sólo espera una. Otra rareza histórica proviene de los días en que no se conocían las unidades de visualización en pantalla de vídeo y la gente se comunicaba con sus máquinas a través de un teletipo (más bien una máquina de télex). Se escribía un mensaje al computador y éste contestaba escribiendo otro. Esto provocaba serios problemas. Si, por ejemplo, se deseaba borrar algunos caracteres del mensaje (quizás una línea del programa) no se podía hacer retroceder la cabeza impresora sobre el papel, como puede hacerse con el cursor en la pantalla. En lugar de esto, se escribían las letras borradas al revés entre trazos verticales para indicar que habían sido eliminadas. Podía por ejemplo preguntarse a la máquina «¿Cómo está tu padre/erdap/madre?». Aunque en el mundo de los computadores nadie utiliza teletipos desde hace años, todavía es posible encontrar lenguajes y sistemas operativos que funcionan de esta manera. Una reliquia aún más insidiosa es el uso continuado de editores que sólo consideran una línea cada vez. Y existen otras incongruencias tan profundamente enraizadas, que nadie sospecha que lo sean.

El problema con los computadores personales hoy día y en los años venideros estriba en que están inmersos en un proceso de selección darwiniana. Uno de los puntos fuertes del sistema capitalista es que cuando existe un problema cuya resolución puede reportar cuantiosos beneficios,

el número de soluciones que se proponen es enorme. Con el tiempo una de estas soluciones, la mejor (o quizá la menos mala), será seleccionada y se convertirá en la estándar. Por desgracia, nosotros, los usuarios, tenemos que hacer en esta especie de proceso de selección el papel de la naturaleza y entresacar las soluciones fallidas. Este trabajo es siempre arduo y en ocasiones puede ser doloroso.

FABRICACIÓN DE CHIPS

La fabricación de chips constituye una mezcla curiosa de la más avanzada tecnología y las técnicas de cocina más primitivas. Cuando se visita una de estas plantas, hay que ponerse un mono de fibra sin costuras para pasar, a través de cámaras con ventiladores y detectores de radiación, a salas increíblemente limpias donde mujeres misteriosamente atractivas, con máscaras y botas de goma, a las que sólo se les pueden ver los ojos, transportan pequeñas bandejas con chips. Las sitúan bajo microscopios y se concentran en las imágenes que aparecen en las pantallas de televisión; van y vienen de los hornos de dopado controlados por computador, transportando pequeñas tarteras de silicio. En esta cocina, el pinche tiene un doctorado; el pepino que se corta en rodajas es un gran cristal de silicio y las secciones se pulen hasta que sean ópticamente mates, hasta grosores de fracciones de micra.

En lugar de una máquina para cortar pan, tienen una sierra de diamante para cortar las secciones en chips. Cuando se han realizado los cortes, del grosor de un cabello, un hombre decidido golpea cada sección con un mazo para que se separen los chips. De las manos de la mujer caen objetos de tamaño muy pequeño; pero no está desvainando guisantes, sino seleccionando procesadores.

La larga fila de trabajadores con microscopios y soldadores parecen operarios de un taller de joyería; pero no, lo que están haciendo es montar los chips en sus fundas y conectar los conductores del grosor de un cabello a las patas de los circuitos integrados.

La señora de aspecto malhumorado se ocupa de una araña mecánica: los circuitos integrados pasan de sus manos a un cesto si han superado las pruebas, y a otro en caso contrario. El encargado del control de calidad está al acecho, ansioso de contar las proporciones, porque, al igual que la

cocina de un restaurante, esta fábrica se gana la vida sirviendo platos de silicio aceptables.

PROGRESOS EN HARDWARE

Los chips de la máquina utilizada para escribir este libro (y casi con toda seguridad los de su microcomputador) están en el nivel de “6 micras”. Esto quiere decir que los conductores utilizados para fabricar los transistores que contienen tienen un grosor de 6 micras (una micra es la millonésima parte del metro). Esta “anchura de línea” es una medida de la mayor importancia, porque nos proporciona inmediatamente gran cantidad de información sobre el rendimiento del propio chip.

En el momento de escribir este libro, los mejores chips existentes en el mercado estaban fabricados con líneas conductoras de cerca de 2,5 micras de ancho. Esto significa que un transistor ocupa un cuadrado de 40 micras de lado y un chip puede contener cerca de 24.000 transistores de este tamaño. Aunque el ojo humano sólo puede ver una pequeña mancha cuando observa un chip de 6 micras, los fabricantes buscan afanosamente conseguir anchuras de línea menores. La razón está (como vimos en las p. 235-237) en que el coste de un chip no se ve afectado por lo que contiene. Si se reduce la anchura de la línea, se obtienen más transistores, más baratos y más rápidos.

De hecho, la mejora del rendimiento es espectacular. Si se reduce la anchura de la línea a la mitad, el número de transistores de un chip se cuadruplica. Además, el número de electrones contenidos en cada conductor se divide también por cuatro, por lo que se mueven a una velocidad cuatro veces mayor. En total, la velocidad queda multiplicada por un factor igual a 16. Sin embargo, la tensión suministrada debe reducirse a la mitad, ya que los conductores se encuentran dos veces más próximos.

El resultado final es que el chip funciona ocho veces más rápido por el mismo precio. Esto tiene gran interés, ya que significa que puede construirse un computador con la misma potencia que antes a un precio ocho veces menor. Además, al vender muchos más computadores (puesto que una de las leyes de marketing dice que si se reduce el precio de un artículo a la mitad, las ventas se multiplican por cuatro), el precio puede hacerse aún más barato, ya que se fabrican muchos más.

El departamento de Defensa de Estados Unidos se ha dado cuenta, como todo el mundo, de los beneficios que comporta la miniaturización, porque ya hace algunos años que estimula a la industria que fabrica chips para que intente con todas sus fuerzas probar anchos de línea pequeños en su programa VLSI (*Very Large-Scale Integrate Circuit;* Circuitos Integrados a Muy Gran Escala). Se han fabricado en gran cantidad chips con anchos de línea de hasta 0,5 micras y si se pudiera comercializar un procesador que utilizase chips como éstos, su potencia equivaldría a veinte mainframes de las series 370 de IBM. Recientemente, Hewlett-Packard ha anunciado un chip procesador de 1 micra y 32 bits que permite colocar una unidad central de un computador literalmente encima de la mesa del usuario.

El principal inconveniente de los anchos de línea pequeños es que con ello resulta mucho más difícil fabricar un buen chip. Un fallo que podría ignorarse en un chip de 5 micras puede inutilizar uno de 2,5 micras. Los fabricantes de chips calculan que se necesita producir al menos un 30% de chips buenos por sección de cristal de silicio para obtener beneficios interesantes.

Parece muy probable que los nuevos chips de alta densidad tendrán un rendimiento de sólo uno bueno por varias secciones de cristal de silicio. Los chips de alta densidad han heredado todo tipo de inconvenientes: las máscaras utilizadas para la fabricación se encogen o se expanden, impidiendo la alineación correcta; partículas de polvo muy pequeñas pasan a través de los filtros más finos y se sitúan entre la máscara y el chip, destruyéndolo; incluso la longitud de onda de los rayos luminosos empleados para imprimir los soportes es demasiado larga, curvándose alrededor de las delgadas líneas, de modo que el fino esquema de líneas degenera en una borrosa con fusión.

Limitaciones de las leyes de la naturaleza

Tal como vimos al inicio de este apartado, si podemos hacer más pequeño el ancho de la línea utilizada para trazar los microcircuitos de un chip, podemos en principio lograr que la máquina funcione a mucha más velocidad. Una máquina más rápida significa dos cosas: o la misma potencia por menos dinero o más potencia por el mismo dinero. Ambas son

(como todo el mundo sabe) una *gran cosa.* Sin embargo, podemos estar seguros que tarde o temprano la naturaleza detendrá este progreso. Es interesante tratar de ver algunos de los límites de la miniaturización.

El primer problema, que incluso ahora preocupa en gran manera, pro viene de que la fabricación de chips es en esencia un proceso de impresión. Varias capas de dibujos muy complicados deben imprimirse una encima de la otra en un espacio muy pequeño. La manera más clara de hacerlo es fotográficamente.

Las máscaras se dibujan a tamaño práctico y luego se reducen fotográficamente a las minúsculas proporciones necesarias para el chip. A continuación se transfiere al chip, revistiendo el sílice con un soporte (un barniz foto sensible) que se expone a la luz a través de la máscara. Las partes expuestas se endurecen; el resto de las máscaras puede eliminarse. Esto permite grabar algunas partes del chip sin afectar a otras; depositar conductores en un lugar y no en otro.

Este sistema funcionaba perfectamente mientras los anchos de línea no eran inferiores a unas pocas micras. Los problemas empiezan cuando el ancho de línea se reduce a menos de una micra, ya que la longitud de onda de la luz visible es aproximadamente igual a la de los diámetros de los conductores. Lo que significa que las líneas dejan de proyectar sombras definidas y los chips empiezan a parecerse al plano de las calles de una ciudad vista a través de una espesa niebla. La solución está en utilizar longitudes de onda más cortas para el proceso de reproducción. Los diseñadores de máquinas para la fabricación de chips pueden usar rayos ultravioleta, con longitudes de onda de una décima a una milésima de micra, o rayos X, con longitudes de onda inferiores a 10^{-14} micras. Pero a medida que las longitudes de onda se acortan, las radiaciones resultan más difíciles de manejar y el proceso fotográfico se complica.

Otra solución es abandonar las máscaras y la luz y pasar a utilizar haces de electrones. Los electrones dotados de la suficiente energía tienen longitudes de onda suficientemente cortas, y pueden ser dirigidos con exactitud mediante campos eléctricos. De hecho, esta tecnología ya se utiliza en los microscopios electrónicos, por lo que pueden fabricarse chips dibujando directamente sobre el silicio un esquema de conductores generado por computador. Este procedimiento funciona bastante bien, pero resulta terriblemente lento, tal como podemos ver fácilmente si recordamos que un moderno chip VLSI puede llegar a tener 16 capas,

cada una de ellas tan compleja como el plano de calles de una gran ciudad. La ventaja del proceso fotográfico tradicional es que se pueden hacer negativos para imprimir 400 chips en una sección del cristal de silicio, repitiendo las máscaras para un chip único, de la misma forma como se imprime una hoja de sellos de correos. En cambio, el sistema del haz de electrones obliga a imprimir cada sello individualmente.

Este procedimiento resulta lento y la velocidad es importante porque lo que se busca al hacer los chips más pequeños es que sean más baratos. Si al final resultan más lentos de fabricar, su precio sube de nuevo. Por otra parte, existen problemas relacionados con los conductores y los electrones en las propias líneas. Al hacerse las líneas más pequeñas, su ancho se aproxima al de los átomos reales del material conductor. No importa demasiado qué tipo de metal se utiliza como conductor, ya que los átomos tienen todos más o menos el mismo tamaño, aproximadamente 10^{-10} metros de diámetro. Cuando el tamaño de un conductor se reduce tanto como para que su ancho equivalga sólo al de algunos átomos, lo más probable es que se evapore antes del tiempo de vida previsto para el computador. Se estima que el ancho razonable de un conductor permanente no puede ser inferior a 20 átomos, lo que sitúa el límite inferior en cinco centésimas de micra: cincuenta veces menor que los chips de hoy en día. Si pudiera fabricarse una máquina que utilizase un ancho de línea como éste sería mil millones de veces más potente que los microcomputadores actuales. Pero antes de que las líneas se hagan tan pequeñas que se evaporen, se presentan otros problemas.

Uno de los más importantes, incluso con líneas de 1 micra, reside en la incertidumbre de la posición de los electrones. Tal como dijo Werner Karl Heisenberg en 1927, no pueden determinarse al mismo tiempo la posición exacta de un electrón y su velocidad. Si se conoce su velocidad (porque deambula por uno de los conductores de un chip ultramicroscópico) no puede saberse dónde se encuentra. Se le supone en uno de los conductores, en un transistor o en cualquier otra parte que su deber en interés de la informática le exija; pero el principio de incertidumbre de Heisenberg dice que en realidad su posición es una cuestión probabilística. Puede estar donde se cree que está, pero también podría estar en cualquier otro lugar; en especial, en un conductor donde no debería estar. Este tipo de situación puede inutilizar el chip mejor diseñado. Se afirma

que efectos probabilísticos como éste ya crean dificultades en las RAM de 64 K.

Una posible solución a este problema sería utilizar cuentas más pequeñas en nuestro ábaco electrónico. El electrón, aunque minúsculo según estándares humanos, resulta pesado e impreciso. Una cuenta mejor podría ser una partícula de luz (un fotón), cuya posición puede determinarse con mayor precisión y no tiene peso. Esto significa que puede ser desviada utilizando menor cantidad de energía, lo que a su vez permite construir computadores más pequeños y que generan menos calor, cosa que, como veremos en seguida, resulta, sin lugar a dudas, de gran interés.

Un computador que utilice fotones podría reemplazar los transistores por láser de semiconductores. Investigadores de la Universidad de Edimburgo disponen desde hace algún tiempo de un dispositivo que permite que un rayo de luz interrumpa a otro, de forma similar a como un transistor hace que una corriente eléctrica interrumpa a otra. Si se consiguieran dispositivos como éste, suficientemente pequeños, podrían utilizarse fotones en lugar de electrones en una estructura igual a la de las máquinas actuales.

Sin embargo, incluso suponiendo que puedan fabricarse chips más pequeños (y conociendo la inventiva humana, no hay duda de que un día se conseguirá) existen otras dificultades. La primera y fundamental se refiere a la velocidad de las señales. Einstein descubrió que nada puede viajar a mayor velocidad que la luz. Esta limitación, que podría parecer poco importante dado que la velocidad de la luz es de 300 millones de metros por segundo, tiene, sin embargo, consecuencias prácticas de gran trascendencia.

Un computador sigue el tiempo marcado por un reloj central; “tic”, se engulle un nuevo bit de datos; “tac”, se procesa; “tic”, los datos siguen su camino. Si queremos que la máquina funcione correctamente, las pulsaciones del reloj deben llegar a todas partes del computador al mismo tiempo, o con una diferencia de un décimo de la pulsación del reloj. Esto significa que la dimensión más grande del computador no puede ser mayor que la distancia que la luz puede recorrer en un décimo de la pulsación del reloj.

Las máquinas actuales trabajan a una velocidad de reloj de 4-10 MHz. Una décima parte de esto es 1/40–1/100 millonésima de segundo. En este tiempo la luz recorre entre 7,5 y 18 m y, sorprendentemente, éste

es el tamaño máximo de una máquina. Si queremos acelerar las cosas aumentando el ritmo del reloj (lo que es una estrategia evidente) tenemos que fabricar máquinas más pequeñas. La anchura de una máquina de 100 MHz no podría ser mayor que 1,5 m. Esto no parece ser un problema si al mismo tiempo se hacen los dispositivos más pequeños; como se deberían hacer si se quiere que funcionen a mayor velocidad. El problema se presenta cuando queremos eliminar el calor desprendido por los millones de transistores que se acumulan en un espacio tan pequeño. Cuando un computador se hace más pequeño y más rápido, también alcanza mayor temperatura. Al final explotará al conectarlo.

La solución más elegante parece ser el computador superconductor, que funciona en un baño de helio. Los dos problemas más importantes se resuelven de golpe: puesto que la corriente eléctrica circula sin resistencia en los superconductores, se desprende poco calor y, debido a la refrigeración necesaria para que la máquina esté a 4º por encima del cero absoluto, el calor que se produzca se elimina con facilidad. Evidentemente, este argumento presupone que deben utilizarse los superconductores actuales; por lo que la necesidad de mantener el computador en un baño de helio líquido, para lograr una temperatura suficientemente baja, es un inconveniente. Sin embargo, parece que hay algunas sustancias (aunque nadie puede afirmar con toda seguridad cuáles) que actuarían como superconductores a temperaturas mucho más altas.

Para construir un computador superconductor podrían utilizarse dos dispositivos. Uno es la conexión Josephson, cuyo funcionamiento depende de dos efectos. El primero es el efecto túnel del electrón. Estamos acostumbrados a que los materiales sean conductores o aislantes eléctricos: el cobre conduce la electricidad; el polietileno no. Como ocurre muy a menudo en física, estos hechos son sólo ciertos cuando se consideran grandes masas de material según estándares atómicos. Si conseguimos hacer una lámina suficientemente pequeña de un material aislante colocado entre dos conductores, un número reducido de electrones practicarán un "túnel" y la atravesarán. De nuevo, la razón de que esto ocurra es esencialmente el principio de incertidumbre de Heisenberg; algunos electrones no están determinados en el lado más alejado de la barrera aislante en el momento crucial y, por tanto, se comportan como si no estuvieran allí. El segundo efecto es que si enfriamos una barrera muy delgada colocada entre dos conductores hasta el punto en que éstos se

convierten en superconductores, la barrera deja de ser aislante. Los electrones circulan sin ningún impedimento a través de ella, hasta que se aplica un campo magnético: entonces se convierte de nuevo en aislante.

En este fenómeno, como indicó Brian Josephson en 1962 (por lo que recibió el premio Nobel diez años después), tenemos los ingredientes necesarios para construir un interruptor electrónico. La corriente que se quiere controlar circula a través de una conexión Josephson; la corriente que ha de controlar esta conexión circula por una bobina próxima que crea un campo magnético. Si la corriente controladora es pequeña o no circula, no hay ningún campo magnético y la corriente de Josephson circula normalmente. Si la corriente controladora aumenta, creando un campo magnético, se llega a un punto en el que la conexión se interrumpe, el aislante se restablece y la corriente controlada deja de circular.

Las conexiones Josephson son interruptores muy rápidos: se “abren” y “cierran” en cerca de 10-15 picosegundos, lo que implica una velocidad de reloj aproximadamente 50.000 millones de ciclos por segundo. Claro que, como vimos anteriormente, un computador que funcione a esta velocidad no podría tener un tamaño superior a 0,06 cm de ancho. De hecho, todo el conjunto debería construirse en un único chip si se quiere que las señales transmitidas a la velocidad de la luz se desplacen de un lado a otro a la rapidez suficiente. Desgraciadamente, las conexiones Josephson ocupan mucho más espacio que los transistores, por lo que su aplicación inmediata no parece viable. Hasta ahora algunas grandes firmas como IBM, han realizado experimentos con estos dispositivos, pero ninguna ha construido un computador comercial con ellos.

Los investigadores de IBM han anunciado un nuevo dispositivo, el Quiteron. Funciona de forma más parecida a un transistor: la tensión aplicada en un terminal conecta o desconecta una corriente superconductora entre los otros dos terminales. Los dispositivos controlados por tensión son mejores que los controlados por corriente, ya que es más fácil enlazar muchos de ellos entre sí. También se dice que el Quiteron ocupa menos espacio en el chip, por lo que es más adecuado para circuitos muy densos.

Procesamiento en paralelo

Sin embargo, ya se empieza a pensar que la solución no está únicamente en lograr procesadores más y más potentes. Muchos de los problemas de la informática continuarían siendo enormes aunque contásemos con procesadores mil veces más rápidos que los mejores que podemos imaginar en la actualidad.

Para entender el porqué, tan sólo tenemos que volver a la descripción que hemos hecho de cómo lograr que un computador "vea" (véanse páginas 174-178). Teníamos que conseguir que el procesador explorase todo el campo visual varias veces, comparando cada pixel con sus vecinos. Los mejores sistemas de visión actuales tienen quizás 100.000 pixeles e, incluso utilizando la potencia de un main-frame, la ejecución de un programa para reconocer a una persona andando puede durar horas. Si lo comparamos con el ojo humano, que tiene unos 3 millones de pixeles (barras y conos) y puede procesar con facilidad todas estas operaciones en 1/25 segundos, nos damos cuenta de lo lejos que estamos de conseguir rendimientos como éste.

En informática, lo que se necesita no es tanto procesadores más rápidos como mayor número de ellos. Después de todo, el ojo humano trabaja de esta manera. La luz incide en barras y conos, que envían señales al cerebro indicando el color y la intensidad de la luz que detectan, y que también hacen entre ellos gran cantidad de procesos de bajo nivel: promediando las interferencias, ajustándose para la luz y sombra, detectando formas simples y movimientos. Lo que se busca en informática es precisamente el diseño de un proceso similar al del ojo humano, al que llamamos "procesamiento en paralelo".

Imaginemos una máquina que tiene procesadores sencillos unidos a cada una de las células sensibles a la luz. La primera etapa en la visión (eliminar las interferencias promediando la señal con la de las células vecinas) podría ser hecha en media docena de ciclos por todas las células procesadoras, frente a los millones de ciclos que necesitaría un procesador central para recorrerlas todas.

De forma similar, todos los procesadores individuales pueden detectar los bordes preguntando a sus vecinos «¿Veis el mismo color e intensidad que yo?». Los que no ven el mismo color e intensidad tienden a encontrarse en o cerca del borde de algún elemento de la escena. El mis-

mo tipo de proceso puede construir simultáneamente regiones de tono similar por toda la imagen. Cuando el procesamiento ha alcanzado el nivel de «¿Es un pájaro?, ¿es un avión?», la misma arquitectura múltiple puede operar como base de datos, buscando muchas identificaciones posibles al mismo tiempo.

Varias firmas están investigando el problema del procesamiento en paralelo. La GEC británica tiene un dispositivo para el procesamiento de la visión llamado chip Grid, en el que varios procesadores se unen entre sí y cada uno a su trocito de memoria, donde puede almacenar los datos y el programa. Para construir una máquina útil, esta memoria debe ser accesible a un procesador central que puede cargar cada elemento con el trozo de programa apropiado: promediar las interferencias; encontrar los bordes; identificar regiones; buscar en la base de datos para ver lo que se tiene...

Hay otro esquema, desarrollado en la Universidad de Stanford, que es mucho menos especializado. Presenta cuatro niveles de procesadores. Cada uno de los procesadores en un nivel controla dos que se encuentran en un nivel inferior, de manera que hay ocho procesadores en el nivel más bajo. Tienen el control de toda la memoria. Como cada procesador puede ponerse en situación de "transparente" (transmitir información sin actuar sobre ella), esta jerarquía proporciona varias posibilidades. En un extremo, el procesador de alto nivel único controla toda la memoria de nivel más bajo y funciona del mismo modo que los procesadores actuales. En el otro extremo, todos los procesadores de alto nivel se hacen transparentes, dejando que los ocho en el nivel más bajo trabajen en paralelo. Y, evidentemente, todas las combinaciones intermedias son igualmente posibles.

La dificultad estriba en que hasta que la gente pueda empezar a jugar con este hardware no sabrá lo que realmente necesita. Y hasta que la gente tenga una idea bastante clara de lo que necesita, nadie, ya sea la administración o la industria, está dispuesto a gastar los miles de millones de dólares necesarios para hacerlo. De momento, aunque pueda intuirse vagamente el hardware que se precisa, nadie sabe exactamente cómo escribir software para controlar unidades de trabajo en paralelo. Resulta bastante fácil escribir software cuando se quiere que muchos procesadores sencillos hagan algo al unísono, como hicimos con la máquina de visión. Pero es mucho más difícil cuando lo que se quiere es

realizar varias tareas diferentes al mismo tiempo, como, por ejemplo, con gran cantidad de información numérica. Un procesador puede estar calculando medias mientras otro calcula variancias. Pero el segundo puede utilizar los resultados del primero para acelerar sus propios cálculos. Ahora se empieza a experimentar con lenguajes de procesamiento en paralelo; sin embargo, nadie ha llegado muy lejos, en parte porque sólo se pueden simular procesamientos en paralelo en máquinas de procesador único. Sin duda no hemos visto un crecimiento explosivo de las técnicas como el que ocurrió cuando el mercado se llenó de computadores convencionales. Por ahora el procesamiento en paralelo es tan sólo un embrión destinado a desarrollarse en el futuro.

La transinformática

Hasta ahora hemos considerado (de forma más bien diletante) las posibilidades de aumentar la potencia y velocidad de los computadores. Resulta interesante considerar el problema desde el otro extremo, preguntándonos: «¿Existen problemas que un computador, sea cual fuere su potencia, no pueda resolver?» Ciertamente los hay. El juego del ajedrez es uno de ellos. Teniendo en cuenta que los computadores acostumbran a ser pedantes, podría pensarse que para lograr que uno de ellos juegue al ajedrez es suficiente explicarle las reglas del juego y dejar que calcule todas las posibilidades de cada una de las jugadas. Una vez hecho esto, debería ser capaz de seleccionar el mejor conjunto de movimientos y seguir ininterrumpidamente esta estrategia hasta la victoria final.

Sin embargo, no es tan sencillo. Como promedio, en una partida de ajedrez se tiene en cada jugada unos 30 posibles movimientos distintos. Cada uno de ellos nos lleva a 30 más en la segunda jugada y cada uno de éstos a 30 más en la tercera. Así, la máquina tiene que explorar 30, 900, 27.000, 24.300.000, ... posiciones, lo que rápidamente se nos escapa de las manos. Resulta fácil darse cuenta de que este juego no podrá ser investigado por completo por ningún computador que se construya próximamente.

En otros muchos problemas de investigación se presenta el mismo tipo de dificultades. Imaginemos que usted es un vendedor que tiene que visitar cien ciudades. ¿Cuál es el orden en que debe visitarlas para reco-

rrer la mínima distancia posible? En la página 208 consideramos las dificultades que se presentan cuando se quiere que un computador "piense" cómo ha de resolver un problema. La única forma de abordar este problema es hacer que la máquina ejecute una de las acciones posibles, luego otra y así sucesivamente, y ver al final si ha alcanzado la solución deseada.

Eternidad significa más tiempo del que ha transcurrido desde el inicio del Universo. Hans J. Bremmermann, hombre con cierta afición a las preguntas sin respuestas, abordó el problema del ajedrez (y el problema del viajante y otros del mismo tipo) del modo como se explica en el siguiente párrafo.

Demostró que el nivel máximo de procesamiento de datos de un computador que pesa m gramos es mc^2/h, donde c es la velocidad de la luz y h la constante de Plank. Esto equivale más o menos a 10^{47} b/s por gramo, lo que parece mucho si se piensa en términos de procesamiento de palabras y no demasiado si se quiere jugar al ajedrez. Para eliminar cualquier argumento sobre lo que un computador podría llegar a pesar, Bremmermann supone simplemente que tenemos uno construido con toda la materia del Universo, que pesa 10^{55} gramos. Se calcula que el Universo existe desde hace 20 mil millones de años, es decir, $6{,}3\times10^{17}$ segundos. Con este tiempo y con un procesador del tamaño anterior no podríamos, probablemente, procesar más de 10^{120} bits. Bremmermann dice que cualquier problema que requiera para su solución más procesamiento de datos que el permitido por esta generosa cifra es "transinformático".

La cifra 10^{120} bits parece considerable; sin embargo, de hecho se alcanza en una docena de movimientos de ajedrez.

Tampoco el Universo actuando como un computador en funcionamiento desde el inicio del tiempo podría resolver el problema del viajante y las cien ciudades. Por otra parte, se cree que muchos de los problemas de la inteligencia artificial serán transinformáticos.

Esto puede parecer deprimente; pero todo lo que nos dice es que la fuerza bruta no es la forma de solucionar los problemas, lo que sabemos perfectamente por experiencia propia, en particular al jugar al ajedrez. Al enfrentarse con una dificultad, sólo la persona más primitiva hará una lista metódica de todas las acciones posibles y luego eliminará las que no tienen valor. Normalmente seguimos la línea que nos conduce al fin

deseado, iluminados las más de las veces por una misteriosa luz que nos dice que vamos por el buen camino. Pero precisamente la forma de programar esta luz ha eludido a las mentes más brillantes.

Quizá usted, amable lector, encontrará la respuesta y se convertirá en el héroe del siglo XX.

ALMACENAMIENTO MASIVO DE DATOS

Los chips más rápidos, más pequeños y más potentes constituyen tan sólo una parte del problema de la informática. Lo que dificulta más que nada el progreso de los computadores personales es la debilidad del soporte de almacenamiento en las máquinas actuales. Hace algunos años se pensaba que 240 kilobytes de datos en un solo disco era mucho más de lo que cualquiera podía soñar; ahora, 10 megabytes (40 veces más) es una cifra bastante común. Sin embargo, incluso esto es demasiado poco comparado con las verdaderas necesidades de almacenamiento.

En un estante de 3,5 m pueden colocarse fácilmente 100 libros, cada uno de ellos conteniendo 50.000 palabras: un almacenamiento de datos de 30 megabytes. Un cajón de un archivador tradicional puede contener 50 archivos, cada uno de ellos con 100 hojas de papel y cada una de éstas con unas 300 palabras: un total de 9 megabytes. Un oficinista llena rápidamente uno de estos cajones. Una oficina con media docena de empleados necesitaría una capacidad de almacenamiento total de 100 a 200 megabytes si prescindiesen del papel en su trabajo diario. De todas maneras necesitarían almacenar a largo plazo archivos, para lo que puede ser más interesante continuar utilizando papel.

Esta clase de argumentos indica la necesidad de dispositivos de soportes de almacenamiento con mucha más capacidad que los discos actuales. Una posibilidad (véanse pp. 62-63) es la grabación magnética vertical. Se asegura que si se magnetiza un disco en regiones parecidas a barras colocadas a través del disco, con una cabeza grabadora a cada lado el almacenamiento podría multiplicarse 40 veces. Lo que significa que se podría disponer de discos Winchester con capacidades de almacenamiento de 1.500 millones de bytes (1,5 gigabytes).

Otra posibilidad más inmediata es el disco láser. Este dispositivo se inventó originalmente para el almacenamiento de programas de televisión que precisan de gran cantidad de datos. Una señal de televisión tiene

una anchura de banda de unos 8 MHz y su codificación digital no puede hacerse en menos de 8 Mb/seg o 1 MB/seg. Es decir, una hora de programa ocupa unos 3.600 megabytes (3,6 gigabytes), más del doble del mejor Winchester posible, cifra que indica precisamente el almacenamiento que debe tener un disco láser para que sea considerado útil.

Un disco láser codifica los datos digitales a través de una serie de pequeños pozos grabados en espiral de forma bastante parecida a un disco gramofónico. Las señales o pozos se leen mediante un láser de baja potencia muy bien ajustado. Como que no hay nada que toque físicamente al disco, éste no se desgasta; además, como el acceso se realiza, haciendo girar el disco y poniendo y sacando el cabezal, esta tecnología podría, en principio, ofrecer la clase de acceso aleatorio que se necesita en informática.

La mayor dificultad hasta hace muy poco consistía en que los discos láser sólo podían leerse. Los datos para grabar un programa de televisión se imprimían en cada disco al confeccionarlo y, por tanto, no podían alterarse. Esto no resultaba demasiado útil en informática; y, tal como demostró la indiferencia de los clientes, tampoco era demasiado adecuado para las grabaciones de vídeo. Sin embargo, un desarrollo reciente parece que puede cambiar esta situación. Algunas compañías han anunciado un nuevo tipo de disco láser que puede grabarse y leerse. El componente esencial de este disco es una película de antimonio-selenio colocada encima de una película de bismuto-telurio, que a su vez está sobre un disco de plástico. Cuando un láser de potencia razonable actúa sobre la superficie, su energía pasa a través de una película de antimonio-selenio para ser absorbida en la capa subyacente, donde se transforma en calor. La estructura amorfa de su capa externa se transforma por la acción del calor en cristalina, lo que la hace reflejante en lugar de mate. En consecuencia, el proceso de grabación produce una señal digital en forma de puntos brillantes sobre fondo gris, que puede leerse mediante un láser de baja potencia.

Uno de los inconvenientes es que los datos no pueden borrarse; pero, puesto que un disco de esta clase puede contener los datos que introduciría una mecanógrafa rápida escribiendo sin parar durante 700 años, este "defecto" resulta cuando menos tolerable. De hecho, como puede decirnos cualquiera a quien un computador díscolo le haya "rayado" una ficha importante, resultaría muy tranquilizante poder tener una copia imborra-

ble de cada uno de los archivos realizados. Sin embargo, Matsushita ha anunciado un dispositivo del mismo tipo en el que el disco puede borrarse y ponerse a punto para volver a grabar aplicando un láser de alta energía.

El punto más problemático al utilizar discos láser en informática es el nivel de errores. Los fabricantes de discos calculan que sus equipos pueden producir un bit equivocado cada 1.000 millones. Las grabaciones para televisión son mucho menos exigentes, ya que un bit equivocado produce un punto en la imagen por 1/25 segundos y el ojo se ajusta fácilmente a esta situación. Sería necesario registrar datos de computador en discos láser repetidas veces, con algún complicado mecanismo de comprobación, para asegurarse de que el nivel de errores se mantiene lo suficientemente bajo.

Otro punto importante, que todavía no se ha convertido en un problema en los computadores personales, es como buscar un dato en un archivo tan grande. La velocidad con que se leen los datos contenidos en un disco está limitada por la velocidad de los chips utilizados para procesarlo y, en última instancia, por las leyes de la física. Imaginemos que buscamos la palabra “hipopótamo” en un disco 4.000 MB lleno. Incluso si se posee uno de los supermicroprocesadores de 32 bits, se tardaría una hora en leer todo el disco, mientras que un computador de 8 bits tardaría un día. Deberemos inventar métodos mucho más sofisticados para clasificar la información almacenada que los utilizados actualmente.

A primera vista, cualquiera de estas tecnologías parece proporcionar mucho más almacenamiento del que nadie, ya sea una persona o un grupo, podría probablemente necesitar. Recordemos, sin embargo, que se necesita codificar los datos varias veces para obtener un nivel de errores aceptable. Además, la gente querrá software que utilice estos discos para guardar dos o tres copias de estados anteriores de sus archivos. Finalmente, parece que los índices necesarios para volver a encontrar un dato ocuparán gran parte del espacio disponible. En definitiva, la mejora no será tan espectacular como podría parecer en un principio.

Sin embargo, existen en perspectiva tecnologías de almacenamiento que arrinconarán definitivamente el problema del volumen.

Una de estas tecnologías utiliza un principio fundamental para los láseres: si se excita un átomo de casi cualquier tipo de sustancia, golpeándolo con luz de longitud de onda apropiada, algunos de sus electrones

exteriores se excitan y saltan, ocupando órbitas más alejadas. Después de cierto tiempo, estos electrones vuelven a su anterior órbita emitiendo un fotón de luz. Esta emisión puede estimularse con un nuevo golpe de láser. Esta segunda emisión es detectada por una célula fotoeléctrica que lee el bit-dato escrito o no por la pulsación del láser original.

Podría constituirse una memoria de disco con dos láseres de baja potencia colocados en ángulo recto, cuyos rayos se sumasen en uno solo de la potencia adecuada únicamente en el pequeño volumen donde se cruzan. Este volumen conforma una celda de datos.

Un láser se apuntaría en dirección al disco y se desplazaría acercándose o alejándose del centro del mismo tal como lo hace el cabezal de escritura/lectura en el funcionamiento de un disco convencional. El otro rayo se dirigiría radialmente hacia el centro y se moverá verticalmente para acceder a las celdas de datos en diferentes niveles. Si los dos láseres se disparasen intensamente al unísono, excitarían los átomos de una celda de datos y escribirían un ‘1’ en ella. Si de nuevo los dos rayos se cruzasen en un lugar determinado, su intensidad sería menor y estimularía una emisión que sería detectada por la célula fotoeléctrica.

EL PUEBLO ELECTRÓNICO

“El pueblo electrónico” es un término acuñado por el gurú de la informática, James Martin, en su libro *The Wired Society*. Lo que quiere decir es que cuando maduren las tendencias hacia hardwares baratos y sistemas amplios de comunicación, la gente podrá usar la electrónica para unirse en una comunidad mundial tan fortuita, íntima e informal como la de un pueblo.

Antes de que esto se haga realidad, habrá que contar con computadores personales más baratos, capaces de visualizar y manipular imágenes de alta calidad en cuatro colores, con suficiente capacidad de almacenamiento de datos para guardar todos los records personales y de negocios de un individuo. También se necesitarán líneas de transmisión de datos de alta capacidad, que puedan enviar grandes cantidades de información a bajo precio. Ciertos servicios postales nacionales ya instalan en estos momentos una red nacional de fibras ópticas que podrá extenderse por todas las viviendas y oficinas y que conectará vía satélite con otras redes en otros países. Como centrales de conmutación se emplearán grandes

computadores, que recogerán mensajes de los buses de datos de alta velocidad, desviándolos a sus destinos.

Dos computadores, estén donde estén, podrán conectarse a través de la red, de manera que dos personas que trabajen juntas (en puntos opuestos del mundo) podrán compartir la misma información. Esta información constará además de los archivos de texto de los ordenadores actuales (véanse pp. 210-215), de fotografías, películas, planos y dibujos dotados de movimiento.

La red puede transmitir, además de textos escritos en el teclado obtenidos de un archivo de datos, sonidos e imágenes, y diagramas en cuatro colores. Estas imágenes pueden ser croquis, planos o dibujos, que los dos usuarios pueden modificar dibujando en el computador. La red puede transmitir también televisión en color; uno de los elementos que una estación de computador deberá incorporar será una cámara de televisión para que los usuarios puedan verse mutuamente. Por otra parte, los usuarios podrán acceder a computadores mayores y bases de datos. La mayor parte de la información almacenada actualmente en las bibliotecas se guardará en grandes computadores accesibles a cualquiera que esté conectado a la red.

Casi todas las personas que tengan que comunicarse con otras utilizarán la red: por ejemplo, los ejecutivos y sus secretarias, los arquitectos y sus clientes, los vendedores y sus compradores, los doctores y sus pacientes, un general y su coronel.

La ventaja del pueblo electrónico es que todas estas personas podrán vivir donde quieran y trabajar juntas como si estuvieran en la misma habitación. Además, el computador actúa en ambas terminales como un asistente personal inteligente, almacenando y recuperando información tanto de su propio banco de datos como de los nacionales.

No hay ninguna razón para que la comunicación a través de la red se limite a dos personas. Los computadores pueden utilizarse como medio de difusión; probablemente, tanto el sonido como la imagen de muchos espectáculos llegarán por las conexiones de datos (véanse pp. 226-229).

Pero como la red puede seleccionar de forma inteligente a la gente que conecta entre sí, se utilizará también como instrumento para actividades sociales y de grupo: escuelas, universidades, clubes, grupos políticos, sindicatos, partidos, colectivos de trabajo, etc. Gracias a la estación de computador doméstica, los niños podrán asistir, con otros niños que

estén a varios cientos de kilómetros de distancia, a clases impartidas por un profesor que podría vivir en otro continente.

Efectos en la sociedad

Como dijo el físico alemán Heisenberg «Predecir es difícil, especialmente sobre el futuro»; pero podemos estar seguros de que el pueblo electrónico tendrá efectos profundos en la organización de los países desarrollados. En la actualidad, cerca de la mitad de los trabajadores en el mundo occidental viven en ciudades y trabajan manipulando información. Se amontonan en las ciudades para pasar los días en oficinas que son, en realidad, enormes archivadores donde se guardan los millones de hojas de papel que producen. La proporción de oficinistas aumentará a medida que las fábricas se automaticen, y una proporción mayor de los recursos nacionales deberá invertirse en trasladar a estas personas desde sus hogares a las ciudades y viceversa.

No hay ninguna razón para que los trabajadores se trasladen físicamente hasta la información cuando podemos trasladar ésta a sus hogares a un coste mucho menor.

Evidentemente, la gente necesitará encontrarse para discutir sus planes y compartir experiencias, pero podrán hacerlo en otros ambientes distintos al de las oficinas.

La mayor parte de su trabajo lo harán desde su hogar, o cerca de él, ya que mucha gente no quiere estar en casa las veinticuatro horas del día. Probablemente, se desplazarán hasta una oficina comunal a poca distancia de sus hogares donde, mientras realizan su trabajo, disfrutarán de la compañía de otros trabajadores. Sin duda, quien quiera retirarse a la naturaleza podrá hacerlo y llevar al mismo tiempo una activa vida profesional. Así, podemos pensar que las redes electrónicas tendrán un profundo efecto sobre las ciudades: reduciendo los desplazamientos diarios, reduciendo los espacios dedicados a oficinas y creando pequeñas comunidades de oficinistas en el campo, cuyos integrantes irán a la ciudad ocasionalmente para encontrarse con sus colegas cara a cara, pero que la mayor parte del tiempo desarrollarán sus tareas rutinarias con el computador.

¿Y EL FUTURO?

Prácticamente desde casi los inicios de la revolución industrial pudo preverse la aparición del teléfono, el barco de vapor, el avión, los vuelos a la Luna y los viajes interplanetarios. El mundo occidental había emprendido el camino del desarrollo y aunque todavía se encontraba lejos de estas cosas su recorrido seguía claramente esta dirección. De forma parecida podemos elucubrar y extrapolar acerca de cómo finalizará el desordenado desarrollo alcanzado por la industria informática.

En primer lugar, la miniaturización de los chips (que significa más potencia a menor precio) continuará mientras el hardware no imponga un límite infranqueable a lo que los computadores pueden hacer. Máquinas suficientemente pequeñas para caber en un bolsillo tendrán la potencia informática de los Cray actuales e incluso más; el almacenamiento de datos permitirá guardar el equivalente a varios millones de volúmenes en un espacio igualmente pequeño. Las líneas de transmisión de datos de alta velocidad ampliarán las grandes bases de datos personales, permitiendo incluir en ellas todo el conocimiento del mundo, clasificado y servido eficazmente por sistemas de software de modo que cualquiera, en cualquier lugar y a cualquier hora, pueda encontrar lo que quiera.

Esta tecnología se aplicará de forma asombrosa en las artes. No parece que exista ningún obstáculo que impida que animación, gestión de base de datos, visión e inteligencia artificial se fusionen constituyendo un nuevo tipo de arte maravilloso que combine cine, novela y juegos de computador. Ofrecido quizás en hologramas, este espectáculo producirá historias en pantallas realistas, de tamaño natural y en tres dimensiones, en las que los personajes se caracterizarán de una forma distinta cada vez. Podrán cambiarse el argumento, la apariencia y la conducta de los personajes ya sea al azar o con la intervención de los espectadores.

Probablemente, la gente podrá introducirse en las historias imaginarias, ya sea como personajes principales o secundarios, según el temperamento de cada cual. Una vez dentro de la historia, la máquina los reproducirá junto a los personajes de ficción y sus decisiones y reacciones influirán el curso de los acontecimientos. Si usted piensa, por ejemplo, que *Lo que el viento se llevó* sería mejor con usted en el papel de Scarlett O'Hara o en el de Rhett Butler, tendrá perfecta libertad para introducirse a sí mismo en el argumento. O, si lo que prefiere son los desastres más

modernos, podrá refugiarse en un mundo devastado en el que se encontrará al último de su raza en un planeta abandonado.

Todo esto abre un amplio campo para el espectáculo y la acción social. Al funcionar a través de una red, nos encontramos que en lugar de una o dos personas que se divierten dirigiendo un guión, habrá ahora mucha más gente actuando en una misma área, que sólo existen en la imaginación del computador. Esto podría reemplazar completamente a los espectadores deportivos, convirtiéndolos en multiusuarios de juegos de hipervídeo interactivos. Si la gente empezara a tomar estas cosas en serio, podrían volcarse al mundo de los negocios o de la política. Si los computadores pueden producir un mundo más interesante y manejable que el mundo real, no hay ninguna razón válida para que nadie tenga que quedarse fuera. No hay ninguna razón que impida que la interacción con el computador no sea mucho más física de la que existe actualmente; así, los corredores, por ejemplo, podrían tener sus propias cintas rodantes conectadas a la red para competir en unos Juegos Olímpicos electrónicos.

Las máquinas dirigirán de forma completamente automática muchos negocios. Los computadores tendrán la capacidad de ver y razonar al menos el nivel de, por ejemplo, un perro pastor. Sin embargo, no es probable que toda esta inteligencia se corresponda con un desarrollo equivalente en la ingeniería de hardware.

Podría muy bien ocurrir que la principal aplicación de estas innovaciones tuvieran lugar en el campo bélico y militares. Quizás a finales del presente siglo existan pocos empleos para soldados. La guerra se habría transformado en una competición simbólica de economías y máquinas.

El desarrollo a gran escala de la inteligencia de las máquinas y de las comunicaciones de alta potencia liberaría de forma eficaz a los usuarios del sistema de la limitación que supone vivir en un lugar determinado. Podría desarrollar del mismo modo su vida intelectual y económica desde casi cualquier parte del globo terrestre. Pero la consecuencia de todo este fenómeno es que las personas que no se adapten a este mundo electrónico (ya sea porque ellos personalmente o las sociedades en que viven no estén a la altura de este reto intelectual) se encontrarán con una desventaja insuperable. Parece muy difícil que este desarrollo no produzca dos mundos: una élite intelectual inmensamente bien servida que controla la información, la política, la guerra y la producción, y una masa rural alejada del mundo de los computadores.

Sin embargo, esto no es nada nuevo. Es el modo como funcionaban todos los pueblos desarrollados hasta el advenimiento de la revolución industrial. Con el desarrollo de máquinas estúpidas que necesitaban un jefe más o menos inteligente y responsable, se produjo la democracia de masas. Parece irónico que la sofisticación técnica nos devuelva a una época que parece reproducir los estados esclavistas del antiguo Egipto o de Roma.

¿ADONDE LLEGAREMOS?

La gran incógnita en el futuro del desarrollo de los computadores estriba en si será posible construir máquinas tanto o más inteligentes que el hombre. Hay quien piensa que antes de un siglo las máquinas serán mucho más rápidas y brillantes que los humanos, que sabrán infinitamente más, trabajarán mucho más deprisa y serán inmunes a las flaquezas humanas, tales como el amor y el odio, que tanto impiden el desarrollo del progreso. Serán capaces de construirse y reproducirse a sí mismas y no necesitarán al hombre para nada. La inteligencia humana habrá evolucionado hacia un nuevo hogar de silicio, desembarazándose de sus limitaciones y dejando a sus primitivos poseedores tan atrás como nosotros hemos dejado a los lagartos.

En un reciente programa de televisión sobre el desarrollo de los computadores se presentaba a un profesor de cabellos grises preguntándose si cuando las máquinas hayan superado totalmente al hombre en capacidad e inteligencia, serán amables con nosotros. Aquí se plantean en realidad dos preguntas. La primera es: ¿puede construirse un computador tanto o más potente que el cerebro humano? La segunda, ¿si pudiese construirse semejante máquina, se asemejaría a una persona o seguirían existiendo diferencias fundamentales?

No es fácil contestar a ninguna de estas dos preguntas. La primera es difícil de contestar porque en realidad no tenemos la menor idea acerca de cómo funciona el cerebro. Está formado por unos cientos de miles de millones de neuronas y parece ser que el trabajo de una neurona tiene relación con el procesamiento de señales, y las señales nerviosas consisten en pulsaciones más o menos rápidas. Las señales se transmiten de neurona a neurona a través de las sinapsis, donde al parecer son transportadas por compuestos químicos raros. Con todo, las neuronas trabajan de

forma parecida a como lo hacen los transistores, pero no tienen una sola entrada y una sola salida sino diez o cincuenta. A pesar de las decenas de años que hace que se investiga el funcionamiento de las neuronas, todavía no se sabe qué hace exactamente una neurona o cómo lo hace. Incluso si llegásemos a entender la función de cada neurona individual, todavía nos faltaría por conocer el modo como se conectan entre sí las numerosísimas neuronas del cerebro para producir la increíble capacidad de procesamiento que poseen los humanos para la visión, el lenguaje y la asociación.

Siempre ha existido la tendencia a explicar el cuerpo y cerebro humanos en términos de la tecnología más avanzada. Cuando yo era niño tenía un libro (no muy moderno) que describía el cuerpo humano como una fábrica, con calderas, pistones y un director con sombrero de copa y levita controlando todo el proceso. Hoy, la moda es ver el cuerpo humano en términos informáticos.

En la página 154 consideramos el cerebro desde el punto de vista de la electrónica. Otra forma posible es comenzar con juegos semejantes al de las «veinte preguntas». La teoría de este pasatiempo es que veinte preguntas de sí o no, bastan a un jugador hábil para adivinar una frase simple, tal como "la oreja derecha del presidente". Dos multiplicado por sí mismo veinte veces es alrededor de un millón, de manera que la popularidad de este juego y el hecho de que el número de preguntas sea veinte y no treinta o quince, sugiere que el cerebro humano puede almacenar alrededor de un millón de ideas. Si tenemos en cuenta que todo el mundo tiene en la memoria más de lo que puede describir exactamente con palabras (el aspecto de sus amigos y de su familia, cómo conducir un coche, acariciar un gato, saborear el vino, el olor de un abrigo), tendremos que encontrar sitio en el cerebro para miles de millones de "ideas". Quizá cada neurona almacena una.

Pero, en principio, no parece haber más limitaciones que las que vienen dadas por nuestra ignorancia para construir una máquina un millón de veces más potente que cualquiera de las que poseemos en la actualidad, que trabaje bajo formas que por el momento somos incapaces de comprender. Esto es todo lo que podemos decir como respuesta a la primera de las preguntas que hemos planteado.

Si es posible conseguir computadores que imiten cualquier función de las que caracterizan la conducta humana, ¿puede considerárseles como

seres vivos? Desde determinado punto de vista todo lo que se comporta de forma similar a como lo hace un ser humano es un ser humano. La persona más parecida a usted podría ser un androide, alguien construido de la forma aparentemente más artificiosa, de carne y hueso, pero que sin embargo sigue siendo una máquina. Si la naturaleza puede obtener por evolución un androide que se asemeja a un ser humano, también puede hacerlo la ciencia de los computadores. ¿O es que existe alguna diferencia esencial entre una máquina y un ser humano?

La pregunta no es nueva. Durante mucho tiempo, quienes creían que los seres vivos no son máquinas, podían señalar inmensas áreas sobre las que nada se sabía y afirmar que detrás se escondía algo que no era en absoluto mecánico. Sin embargo, a medida que nuestro conocimiento de la naturaleza animada e inanimada progresa, se hace cada vez más difícil creer que la ciencia en su investigación del ser humano encontrará un punto a partir del cual las leyes de la naturaleza dejan de tener validez. Y si no existe ninguna estructura particular que distinga a los hombres de las máquinas, la diferencia debe buscarse en otra parte.

Hay quien ha propuesto la idea de que el libre albedrío tiene su fundamento en la aleatoriedad del mundo subatómico. Quizás el cerebro es únicamente un gran amplificador que lleva la “vida” del nivel subatómico hasta la escala humana. Esta “vida” que nosotros percibimos como aleatoria e imprevisible, podría reflejar las leyes y los acontecimientos que ocurren en un Universo situado “en ángulo recto” en relación al nuestro que nos es imposible ver. Pero incluso si esto fuera así no existe ninguna razón para preferir las neuronas a los transistores como vías de paso de esta conducta aleatoria. Por otra parte, la “vida” se basa en la inmensa complejidad de los organismos; entre cientos de miles de millones de neuronas, nosotros, por el momento, todo lo que podemos hacer es acumular megabytes de RAM.

Una tercera hipótesis es que las leyes de la física, que parecen tan inexorables, son semejantes a las leyes estadísticas que gobiernan la distribución de las diferentes letras en esta página; tal distribución es accidental y tiene muy poco que ver con el verdadero significado del texto.

Pero en última instancia y por lo que sabemos hasta ahora, el cerebro consiste en moléculas ordinarias dispuestas de formas complejas que obedecen a leyes lógicas. Lo que actualmente sabemos sobre computado-

res nos permite afirmar que si comprendiésemos estas leyes podríamos construir una máquina que las obedeciese. Y si la construimos, lo que tendríamos entonces sería probablemente un computador vivo. Si esto sería bueno o malo es algo difícil de decir, pero realmente no es un problema que deba preocuparnos en esta década, ni siquiera en el presente siglo.

Apéndice 1

INSTRUCCIONES DEL BASIC

Guía simplificada del lenguaje BASIC basada en el MBASIC 80 de Micro soft.

Operadores

A = B	hace A igual a B
C↑D	eleva C a la potencia D
– X	hace X negativo
K + L	suma K a L
M – N	resta N de M
E•F	multiplica E por F
G/H	divide G por H
P$ + R$	añade R$ al final de P$

Los operadores relaciónales comprueban dos valores: la expresión se reemplaza por –1 si es verdadera, por 0 si no lo es. Pruebe: PRINT (1 = 1). Debería obtener –1. PRINT (1 = 2) debería obtener 0.

¿Por qué PRINT (A - B) produce –1?

A = B	A igual a B
A <> B	A no es igual a B
A > B	A es mayor que B
A< B	A es menor que B
A >= B	A es mayor o igual que B
A <= B	A es menor o igual que B

Los operadores lógicos comparan dos bytes de bit en bit (véanse pp. 14-15): NOT; AND; OR; XOR

Instrucciones del BASIC

AUTO (m,n): Se utiliza cuando se entran programas. Numera cada nueva línea escrita desde m, aumentada en n. Si se omiten m y n, ambas valen 10. CALL (nombre variable) (conjunto de argumentos): Llama una subrutina en lenguaje máquina a la dirección (nombre variable) y le pasa los argumentos.

CHAIN (nombre del archivo): Carga y ejecuta el programa en él (nombre del archivo).

CLEAR m,n: Pone todas las variables en 0 o nulo. Si m y n están presentes, pone la ubicación de memoria más alta utilizada por el BASIC en m (de manera que el código en lenguaje máquina pueda cargarse por encima de ella) y la pila *(stack)* deja un espacio hasta n.

CLOAD (nombre de archivo): Carga y ejecuta (nombre de archivo) desde la cinta cassette.

CLOSE # m, # n...: Cierra las fichas numeradas m, n... en una sentencia OPEN.

COMMON (lista de variables): Pasa las variables a un programa encadenado (CHAINed).

CONT: Para continuar la ejecución después de t C para interrumpir, STOP o END. Se utiliza para la eliminación de errores.

CSAVE (expresión alfanumérica): Guardar el programa que se está ejecutando llamado (expresión alfanumérica) en la cinta.

DATA, m,n,o,p...: Proporciona una lista de constantes, que podría ser una serie de caracteres (string). Se transfieren a las variables de programa con REA D a,b,c...

DEF FN(nombre) (lista de parámetros) - (función): Para definir sus propias funciones. Por ejemplo: DEF FNADD (X,Y) = X + Y. Más tarde en el programa puede escribir Z = FN(A,B). Z se convierte en la suma de A y B.

DEFINT/SNG/DBL/STR (rango de las letras): Define las variables que empiezan con el rango de las letras como números enteros de precisión sencilla o doble.

DEF USR (dígito) = (expresión entera): Especifica la dirección de inicio de una subrutina de lenguaje simbólico de programación (*assembly*). (CALL es mucho mejor.)

DELETE m-n: Borra las líneas de programa desde m hasta n.

DIM (lista de matrices): Especifica el tamaño máximo de las matrices de números o de caracteres.

EDIT n: Edita la línea número n. El programador puede mover el cursor, introducir, anular, encontrar y reemplazar textos en la línea.

END: Detiene el programa, cierra todos los archivos y queda a la espera de nuevas instrucciones.

ERASE (lista de matrices variables): Elimina las matrices en la lista y libera la memoria que ocupan.

FlELD#n,p AS R$, r AS T$...: Fija campos de datos en un archivo aleatorio (véase más abajo). El primer campo es R$ y tiene una longitud de p bytes, el segundo es T$ y tiene una longitud de r bytes, etc...

FORK = nTOr (STEP p)... NEXT K: Ejecuta las líneas del programa entre las dos sentencias, añadiendo p a K cada vez hasta que se hace igual a r. Si se omite “STEP p”, el incremento es 1.

GET#n, p: Lee el record p de la ficha n en la estructura preparada la orden FIELD apropiada.

GOSUB n... RETURN: Transfiere la ejecución del programa a la línea n. Cuando se llega al RETURN, la ejecución vuelve a la sentencia ubicada después de GOSUB.

GOTO m: Transfiere la ejecución a la línea m IF ... THEN ... ELSE: Valora la expresión situada después de IF. Si pasa a –1 (véase los operadores de relación al principio), se ejecutan las instrucciones indicadas después de THEN. Si no, se ejecutan las instrucciones después de ELSE. Si no hay ELSE, la ejecución “cae” a la próxima sentencia del programa. Sin embargo, una sentencia en la misma línea —IF ... THEN (sentencia) será ejecutada sólo si la prueba dio *resultado positivo.* Esto no es lógico pero resulta útil.

INPUT "prompt"; A,B...; Imprime el "prompt" si está presente y espera a que el usuario escriba las variables A,B, etc... separadas por comas. Puede usarse con series de caracteres o números.

lNPUT#n,A,B...: Igual que antes, pero obtiene las variables del archivo n. KILL (nombre del archivo): Borra el nombre del archivo del disco en uso. LET A = B: Hace A igual a B ("LET" es optativo).

LINE INPUT "prompt"; A$: Obtiene una línea completa con comas, comillas, etc.

LINE INPUT#n,A$: Igual que antes a partir del disco.

LIST m-n: Lista las líneas de programa de m a n.

LLIST m-n: Imprime las líneas de m a n en la impresora.

LOAD (nombre del archivo): Carga el archivo a partir del disco.

LPRINT USING "##,##"; A,B...: Imprime las variables A,B, etc., en la impresora dándoles el formato cuando la expresión USING está presente. La impresión de estas variables produciría el siguiente resultado:

23,456	23,45
1,6	1,6
100	0.0 (con un error de superación de capacidad).

LSET (serie de caracteres) ... RSET(serie de caracteres): Introduce una variable en un campo preparatorio para PUT para escribirla en un archivo al azar.

MERGE (nombre del archivo): Lee el archivo a partir del disco (que debe estar en ASCII — véase SAVE) y lo intercala en el programa en curso.

MID$(A$,m,n) = B$: Toma n caracteres de B$ y los escribe sobre A$ mirando a la emésima, por ejemplo: MID$ ("MOJIGATOS",5,5) = "PERRO" produce "MOJIPERRO".

NAME (antiguo nombre del archivo) AS (nuevo nombre del archivo): Da nuevo nombre al archivo.

NEW: Borra el programa en la memoria.

ON ERROR GOTO m: Una especie de GOSUB. Cuando se produce un error, la ejecución va a la línea m. El número error se escribe sobre la variable ERR, y la línea en que se produce sobre ERL. La subrutina en la línea m puede comprobar el error, hacer algo útil y RESUME la ejecución en la línea apropiada.

ON A GOSUB/GOTO m,n,o,p...: Valora A con un dígito r y va al errésimo número de la línea en la lista m,n,o,p,...

OPEN “m”,#n, (nombre del archivo): Abre el archivo y le da el número n. El modo m puede ser “I” por un input serial para el programa. “O” por un output. “R” por al azar *(random).*

OUT i,j: Envía el entero i a la salida j.

POKE i,j.: Mete el entero i en la dirección de memoria j.

?/PRINT USING “exp”, a,b,c,...: Imprime una lista de variables en la pantalla. “?” es una abreviatura de PRINT. Véase LPRINT for USING. Si las variables están separadas por comas, están etiquetadas; si están separadas por puntos y comas, pasan adelante. Al final se imprime una nueva línea a menos que haya una coma o un punto y coma.

PRINT #n,a,b,c,...: Imprime las variables para el número de archivo n.

PUT#n,K: Escribe los datos en el campo apropiado establecido por LSET o RSET, para el record kaésimo en el número de archivo al azar n.

RANDOMIZE (n): Reinicia el proceso de aleatorización con el número n como simiente. Si la simiente no se cambia, se obtendrán los mismos números aleatorios. Si se omite n, el programa se parará y pedirá una simiente al teclado.

READ a,b,c,... Transfiere los próximos ítems de una sentencia DATA a las variables a,b,c,... Véase RESTORE.

REM: Lo que sigue es una observación.

RENUM m,n,o: Numera de nuevo las líneas de programa para empezar con m, desde n en los números antiguos, incrementado por o.

RESTORE m: Hace que el READ siguiente mire la sentencia DATA en la línea m.

RUN m: Inicia la ejecución del programa en la línea m. Si se ha omitido m, lo ejecuta desde el principio.

SAVE (nombre del archivo), A: Guarda el programa en curso bajo el “nombre del archivo”. Si está seguida por “A”, el archivo se guarda en formato de texto. Puede entonces corregirse, compilarse o intercalarse.

STOP: Para el programa. CONT reanuda su ejecución.

SWAP A,B, A$, B$: Intercambia las dos series de caracteres o variables numéricas.

TRON/TROFF (para la eliminación de errores): Imprime los números de las líneas mientras el programa las ejecuta. TROFF anula esta orden.

WHILE (exp)... WEND: Si la expresión después de WHILE es verdadera, se ejecutan las líneas de programa hasta WEND. Como un bucle FOR ... NEXT.

WRITE (#n): Como PRINT, pero pone comillas en las series de caracteres y comas entre los ítems.

Funciones sobre una variable

ABS(X): Valor absoluto de la expresión X: PRINT.

ABS (7•[–5]) daría 35.

ASC (X$): Valor ASCII del carácter en X$.

ATN(X): Arcotangente de X (en radianes).

CHR$(I): Caracter cuyo código ASCII es I.

COS(X): Coseno de X (en radianes).

EXP(X): Elevado a la potencia X.

FRE(X): Cantidad de memoria libre.

HEX$(X): Valor hexadecimal de un número decimal X, por ejemplo, HEX$(32) = 20.

INPUT$(m,n,): Siguientes m caracteres escritos en el teclado, o, si está presente n, del archivo n.

INSTR(i,A$,B$): Busca la primera vez que B$ se presenta en A$ (empezando en el iésimo carácter si está presente i) y da el número del carácter en que esto ocurre. Por ejem.: INSTR(4,"Un romance de verano"," ") dará 11.

INT(X): Mayor entero menor que X.

LEFT$(A$,i): i caracteres más a la izquierda de A$.

LEN(A$): Longitud en caracteres de A$.

INFORMÁTICA PARA TODOS

LOG(X): Logaritmo neperiano de X.

MID$(A$,i,j): Serie de j caracteres de longitud que empieza en el iésimo de A$.

OCT$(A): Valor octodecimal de un número decimal A.

PEEK(i): Byte almacenado en la ubicación de memoria i.

RIGHTS($,i): i caracteres más a la derecha de A.

RND(X): Próximo número aleatorio en la sentencia. X es una viable muda.

SGN(X): Signo de X.	Si X > 0	SGN(X) = 1
	Si X = 0	SGN(X) = 0
	Si X < 0	SGN(X) = –1

SIN(X): Seno de X (en radianes).

SPACE$(X): X espacios.

SQR(X): Raíz cuadrada de X.

STR$(X): Convierte el número X en un string.

STRING$(i,j): una serie de caracteres de longitud i, que empieza en el primer carácter de J$.

TAB(I): Desplaza el cursor I caracteres.

TAN(X): Tangente de X (en radianes).

VAL(X$): Convierte X$ en un valor numérico.

Apéndice 2

LISTADOS DE PROGRAMAS

Estos programas han sido seleccionados entre muchos otros publicados en la revista británica *Practical Computing* durante el período en que el autor de este libro fue su editor. Agradezco a IPC Electrical and Electronic Press Ltd su autorización para reproducirlos aquí. Se han escogido de manera que ilustren diversas técnicas básicas y presenten cierta variedad. He eliminado los programas que exigían conocimientos muy específicos en el manejo de la pantalla.

La mayoría de los listados mejoraría insertando la instrucción de borrado de la pantalla de su máquina.

Código

Puesto que la historia de la informática se inició con la resolución de códigos secretos, parece razonable dar un pequeño programa para escribir mensajes en clave y para descifrarlos (aunque no se haga ilusiones acerca de la posibilidad de engañar a los servicios de inteligencia y a los organismos responsables de la seguridad nacional).

La escritura en clave es en el fondo muy sencilla. Se toma el mensaje letra por letra y se transforma en números (los códigos ASCII son perfectamente válidos). Luego se cambia cada número mediante un sistema que sólo usted y la persona que debe leer el mensaje conocen. La forma más sencilla de hacerlo es fijar una lista de números aleatorios y sumarlos a los números correspondientes al código de letras. Éste es el principio del *one time pad* que proporciona un esquema de codificación irresoluble. El inconveniente está en que la lista de números aleatorios debe ser tan larga como la longitud total de todos los mensajes que se quieren intercambiar. Los criptólogos de la Segunda Guerra Mundial obtuvieron sus resultados mediante la cuidadosa correlación de cada trozo de información que podían captar, lo que significaba decodificar grandes cantidades de mensajes acerca de los temas más pedestres.

Por el contrario, los ejércitos y diplomáticos modernos tienen que poner en clave grandes cantidades de material para burlar un análisis de este tipo y, por consiguiente, tendrían que distribuir montañas de archivos

antiguos y de un solo uso para utilizar esta codificación. Todo esto no resulta demasiado práctico, por lo que se busca un modo de generar una lista de números aleatorios que sean lo bastante aleatorios para que el enemigo no pueda preverla, pero que los amigos puedan reproducir. No es fácil hacerlo, ya que cualquier proceso mecánico tarde o temprano generará la misma lista de nuevo. Cuando esto ocurre, el descifrador tiene la oportunidad de encontrar la solución. Además, también puede aprovechar la posible existencia de algunas regularidades en la lista.

Un modo sencillo de generar una lista de números (una clave) lo bastante aleatoria para que no sea fácil de descifrar desde el exterior, pero lo bastante regular para que puedan recordar los amigos, consiste en utilizar una palabra clave. Cada letra de la palabra clave se emplea para codificar la letra correspondiente del mensaje. Si la palabra clave es más corta que el mensaje, se recicla. Esto es lo que realiza el programa siguiente:

```
10 '******CODE2******
20 KEY$="ZEBRA"
30 INPUT" Caracteres a codificar" ;I$
40 FOR K=1 TO LEN (I$)
50 L=L+1
60 IF L>LEN(KEY$) THEN L=1
70 K$=MID$(KEY$,L, 1) 'GET THE L'TH LETTER OF KEY$
80 J$=MID$(I$,K,1)
90 O=ASC(K$) XOR ASC(J$)
100 PRINT O;" ";
110 NEXT K
```

El DECODIFICADOR 2 efectúa la decodificación:

```
10 '*****DECODE2******
20 KEY$=" ZEBRA"
25 INPUT "Longitud del código" ;LC
40 FOR K=1 TO LC
50 L=L+1
60 IF L>LEN(KEY$) THEN L=1
70 K$=MID$(KEY$,L,1) 'GET THE L'TH LETTER OF KEY$
80 INPUT"Siguiente número de código";J
85 LPRINT J;"
```

```
90 O=ASC(K$) XOR J
100 LPRINT CHR$(O);
110 NEXT K
```

Ambos programas podrían modificarse fácilmente de manera que la palabra clave entrase cuando se ejecutaran.

Este sencillo esquema no presentaría muchas dificultades a un decodificador competente, ya que la clave se repite al final de la palabra clave. Puede alargarse la repetición volviendo a cifrar la salida con una segunda palabra clave que haría que la repetición tuviera la longitud de las dos palabras multiplicadas entre sí. El proceso seguiría utilizando tantas palabras cifradas como se deseen: seis palabras de ocho letras cada una daría una repetición de 8^6: más de 250.000 caracteres. Si se tomó la precaución de cambiar las palabras claves antes de enviar todo este texto, su nivel de seguridad sería bastante elevado. El usuario profesional de códigos fracasa debido a la necesidad de enviar millones de caracteres utilizando la misma clave. Las claves más sencillas continuarían siendo mejores que ésta porque en ellas el criptoanalista se enfrenta al problema sin la ayuda de las regularidades existentes en las palabras claves.

Anagrama

Estos programas tienden a centrarse en la manipulación de textos, porque esto es precisamente lo que los computadores realizan con mayor facilidad. Todo el mundo se ha enfrentado alguna vez a un anagrama. El siguiente programa toma una serie de letras que serán transpuestas: luego, pregunta si se conocen algunas letras en la salida e imprime a continuación todas las posibilidades. En este ejemplo, las letras que deben disponerse de otro modo son “AARAS”, y las letras conocidas son -N-G—M-. Los resultados se presentan a continuación:

```
100 'ANAGRAM PROGRAM
110 'M G PRITCHARD, PC APRIL 1980
120 PRINT "ANAGRAM"
130 PRINT "————————"
140 INPUT"Teclee solo las letras que hay que ordenar"; A$
150 PRINT
```

```
160 L=LEN(A$)
170 INPUT"Hay algunas letras conocidas? (Y/N)";Q$
180 IF Q$="N" OR Q$="n" THEN 240
190 IF Q$<>"Y" AND Q$<> "y" THEN 170
200 PRINT :PRINT:INPUT' Teclee las letras conocidas ej. '–B––D–
    '";K$:W=L
210 FOR J=1 TO LEN(K$)
203 IF MID$(K$ J,1)="–" THEN T=T+1
205 IF LEN(K$)< >L THEN PRINT"Longitud incorrecta. Inténtelo otra
    vez":GOTO 200
206 NEXT J
210 T=0:FOR J=1 TO LEN (K$)
212 IF MID$(K$,J,1)—" THEN T=T+1
215NEXT J
230 GOTO 270
240 K$=" ":FOR J=1 TO L:K$=K$+"–": NEXT J
250 PRINT:INPUT"Número de letras inicial";W
260 IF W<1 OR W>L OR W<>INT(W) THEN PRINT"Error":GOTO
    250
270 DIM B$(L),C$(L),Q(L)
280 PRINT:PRINT
290 GOSUB 5000
300 FOR J=W TO L
310 K=1
320 Q(K)=1
330 IF B$(Q(K))="" THEN 440
340 C$(K)=B$(Q(K)):B$(Q(K))=""
350 K=K+1
360 IF K<J THEN 320
370 A=1
380 FOR S=1 TO LEN (K$)
390 IF MID$(K$,S,1)="–"THEN PRINT C$(A);:A=A+1: GOTO 410
400 PRINT MID$(K$,S,1);
410 NEXT S:PRINT,
420 K=J
430 B$(Q(K))=MID$(A$,Q(K),1)
440 Q(K)=Q(K)+1
```

```
450 IF Q(K)<=L THEN 330
460 K=K–1
470 IF K>=1 THEN 430
480 NEXT J
490 END
5000 FOR N=1 TO L
5010 B$(N)=MID$(A$,N, 1)
5020 NEXT N
5030 RETURN
```

Cloze

Este programa tiene su origen en la investigación para la enseñanza del inglés. Permite al profesor introducir un texto en las líneas 100-200. El profesor o profesora puede escribir:

WHEN I CAME BACK FROM MY HOLIDAY, I
RAN STRAIGHT OUT INTO THE GARDEN TO
SEE MY RABBIT WILLIAM

Entonces, Cloze pide el número de palabras que debe saltarse antes de imprimir un conjunto de rayas. Si el profesor dice dos, borraría la pantalla y escribiría:

WHEN I — BACK FROM — HOLIDAY, I —
STRAIGHT OUT — THE GARDEN — SEE MY
WILLIAM

A continuación el programa pide al alumno que introduzca de una en una las palabras que faltan. Si la primera se corresponde con la adecuada, la máquina imprime la frase de nuevo hasta el segundo espacio y pide al alumno que introduzca otra palabra.

Desde un punto de vista técnico, su interés reside principalmente en la mejora de la instrucción INPUT de BASIC en las líneas 100-200. Las letras se toman del teclado de una en una con I$=INPUT$(1) y se añaden a la palabra admitida con la línea 180. Esto significa que el programador tiene que hacer frente a anulaciones (línea 130), pero también significa que puede comprobar el input para un carácter de “fin de entrada”, en

este caso '*', y actuar en consecuencia. En un programa más sofisticado, que requiere el pleno control del cursor en la pantalla, este tipo de input puede determinar los caracteres utilizados para mover el cursor arriba, de lado a lado y abajo, e imprimir el código apropiado en la pantalla. Por ejemplo, podríamos querer que los usuarios del programa escribiesen CONTROL D (^D) para mover el cursor hacia abajo. Cuando la rutina de input detectara ASCII 4, entraría en una subrutina y PRINT CHR$(8), lo que haría que el cursor descendiese una línea. Una mejora a este esquema consistiría en comparar la línea 310 con las letras mayúsculas y las minúsculas.

Vale la pena esforzarse para que se comprenda la forma como, en las líneas 270-277, el programa Cloze resuelve el problema de imprimir con precisión las rayas correspondientes a las letras de las palabras suprimidas.

```
10 GOSUB 1000
15 M=0
20 PRINT"CLOZE"
30 'REM AUTOR CHRIS HARRISON, PC JUNE 1982
40 FOR I=1 TO 1000:NEXT
49 'REM A$ = PALABRAS DEL TEXTO; R$= RESPUESTA DE LOS
   ALUMNOS SOBRE EL PRIMER ESPACIO DISPONIBLE
50 DIM A$(200), R$(200)
59 'REM ES NECESARIO UN ASTERISCO PARA DETERMINAR EL
   FINAL
60 PRINT "ESCRIBA SU TEXTO AQUÍ SIN COMAS CUANDO
   HAYA TERMINADO INTRODUZCA UN ESPACIO Y UN
   ASTERISCO
80 M=M+1:N=1 'REM N PONE LAS LETRAS EN EL CONTADOR
   DE LA PALABRA
90 'REM EL TEXTO ES INTRODUCIDO LETRA A LETRA
100 I$=INPUT$(1) 'REM E EQUIVALENT TO INKE Y$ OR GET$
120 'REM AHORA PERMITIMOS EL BORRADO. SI N<1
   EMPEZAMOS LA PALABRA DE NUEVO. SI SE UTILIZA LA
   TECLA DE RETROCESO TIRA ATRAS EL CURSOR E IGNORA
   LA LETRA ANTERIOR
```

```
130 IF N<1 THEN 90 ELSE IF I$=CHR$(?) THEN
    A$(M)=LEFT$(A$(M),N–2)):N=N–1 :PRINT CHR$(8):GOTO 100
140 IF I$="" THEN 100
149 'REM EL ESPACIO NO CUENTA COMO PALABRA
150 N=N+1:IF I$=" " THEN PRINT" ";:GOTO80
159 'REM IF I$="*" FIN DEL TEXTO
160 IF I$="*" THEN 210
169 'REM IF SI SE PULSA RETURN NO SE HACE NADA
170 IF I$ = CHR$(13) THEN 100
179 'REM CADA PALABRA ESTA YA CONSTRUIDA
180 A$(M)=A$(M)+I$
189 'REM AHORA IMPRIME CADA LETRA
190 PRINT I$;
199 'REM REPITE EL PROCESO
200 GOTO 100
209 'REM S ES USADA COMO ESPACIO
210 PRINT:PRINT:PRINT:INPUT" ¿QUE INTERVALO DESEA?";S
219 'REM 3 REPRESENTA 2 PALABRAS Y UN BLANCO
220 S=S+1
229 'REM YA ESTAMOS PREPARADOS. LIMPIAMOS LA
    PANTALLA EN LA LINEA 240
240 GOSUB 1000
269 'REM AHORA IMPRIMIMOS UN ESPACIO ANTES DE CADA
    PALABRA, IMPRIMIMOS LAS PALABRAS COMO TANTOS
    GUIONES COMO LETRAS TIENEN
270 PRINT" ";:FOR 1=1 TO M STEP S
272 FOR J=l TO I+S–1
275 PRINT A$G);" ";:NEXT J
277 PRINT STRING$ (LEN(A$0)),"–");" ";:I=I+1:NEXT I
279 'REM AUTORIZA TANTAS RESPUESTAS COMO BLANCOS
    HAY
280 FOR I=S+1 TO M STEP S+l
299 'REM PON EN BLANCO LA PARTE SUPERIOR DE LA
    PANTALLA. LA USAREMOS PARA MENSAJES Y AUTORIZA
    RESPUESTAS
300 PRINT"LLENE LOS BLANCOS";:INPUTR$(I)
309 'REM ¿ES CORRECTA LA RESPUESTA?
```

```
310 IF R$(I)=A$(I) THEN 350
325 IFV>1 THEN PRINT"MALA SUERTE. LA PALABRA ES ";A$(I);
    ELSE 330
326 FOR L=1 TO 300:NEXT L:GOTO 350
329 'REM SI RESPUESTA ERRONEA DALE OTRA OPORTUNIDAD
330 PRINT"NO, MALA SUERTE. PRUEBE DE
    NUEVO";V=V+l:GOTO 300
349 'REM SI LA RESPUESTA ES CORRECTA, IMPRIME EL TEXTO
    ORIGINAL HASTA EL PUNTO ALCANZADO
350 FOR K=1 TO I:PRINT A$(K);" ";:NEXT K:PRINT
370 V=0:NEXT I
380 PRINT"TODA LA FRASE ERA":FOR I=1 TO M:
PRINT A$(I);" ";:NEXT I:PRINT
389 'REM PERMITE UN NUEVO PROCESO
390 PRINT"¿OTRA VEZ? (Y/N)";:INPUT R$
410 PRINT"FIN"
1000 'REM VIA RAPIDA PARA LIMPIAR LA PANTALLA
1010 FOR K=1 TO 24:PRINT:NEXT K
1020 RETURN
```

Vida

Un computador no hace nada que no pueda hacerse con lápiz y papel y tiempo suficiente; pero puede hacerlo a veces tan rápido que parece tomar vida propia. Un hermoso ejemplo de esta cualidad es el juego de “Life” (vida), inventado por el matemático británico John Conway[12]. No es un juego en el sentido corriente, porque, una vez que lo hemos puesto en acción, “juega” por sí solo; sin embargo, no por esto deja de ser fascinante.

La idea es extremadamente simple. El juego se realiza sobre un gran tablero de cuadros, contenido en el computador como bits o bytes de memoria. Cada cuadrado puede estar vacío o contener una célula “viva”; es decir, 0 o 1. Tiene ocho vecinos:

[12] Elwyn R. Berlekamp, John H. Conway, Richard K. Guy, *Winning* Ways, Academic Press, 1982.

0	0	0
0	X	0
0	0	0

El computador examina en cada jugada cada uno de los cuadros del tablero, aplicando las siguientes reglas:

una célula muerta vuelve a la vida si tiene exactamente *tres* vecinos vivos (estas células tienen tres sexos y no dos);
una célula que tiene cuatro o más vecinos vivos muere ahogada;
una célula con sólo un vecino vivo -o ninguno- muere por abandono.

El jugador tan sólo tiene que poner en marcha el juego con una determinada disposición inicial de las células y observar asombrado cómo aparece la “vida”.

```
10 W=40:H=24:CLR$==CHR$(29)'REM LA ANCHURA, ALTURA Y
    "HOME" DE SU PANTALLA
15 W=W–1 :H=H–1
20 DIM A(W+2,H+2),B(W+2,H+2)
30 'REM INPUT EMPIEZA EL CAMINO TERMINADO MEDIANTE
    ""
40 PRINT'Input líneas de celdas. Pulse RETURN para terminar"
45 L=0
50 INPUT A$:L=L+1
60 IF A$="" THEN 200
65 IF ML<LEN(A$) THEN ML=LEN(A$) 'GET LON GEST LINE
70 IF LEN(A$)>W THEN PRINT"muy largo":GOTO 50
80 IF L>H THEN 200
90 FOR K=1 TO LEN(A$)
100 C$=MID$(A$,K,1):IF C$<>" " THEN A(K,L)=1
110 NEXT K
115 GOTO 50
200 'REM, AHORA EL SOFTWARE EMPIEZA EL CAMINO DESDE
    LA MITAD
210 SW=INT((W–ML–1)/2)
220 SH=INT(H–L–1)/2
230 FOR K=1 TO ML
```

```
240 FORJ=l TO L
250 B(K+SWJ+SH)=A(K,J)
260 NEXT J
270 NEXT K
280 GOSUB 10000
300 'REM AHORA GENERAR A PARTIENDO DE B
310 POP=0
320 FOR K=1 TO H
330 FOR J=l TO W
340 N=0:C=B(J,K)'REM CONTADOR DE VECINOS A CERO,
    GUARDAR EL VALOR DE LA CELDA
370 N=B(J–1,K–1)+B(J,K–1)+B(J+1 ,K–1)+B(J–1,K)+B(J+1,K)+B(J–
    1,K+1)+B(J,K+1)+B(J+1,K+1)'REM CONTAR VECINOS
410 GOSUB 30000
420 A(J,K)=NXT:POP=POP+1 'REM PONER LA SIGUIENTE
    GENERACION EN A
430 NEXT J
440 NEXT K
450 IF POP=0 THEN STOP
460 GOSUB 20000
490 'REM PONER LA GENERACION EN B
500 POP=0
510 FOR K=1 TO H
520 FOR J=l TO W
530 N=0:C=B(J,K)
540 N=A(J–1,K–1),+A(J,K–1)+A(J+1,K–1)+A(J–1,K)+A(J+1,K)+A(J–1
    ,K+1)+A(J,K+1)+A(J+1,K+1)
550 GOSUB 30000
555 B(J,K)=NXT:POP=POP+1
560 NEXT J
570 NEXT K
580 IF POP=0 THEN STOP
590 GOSUB 10000
600 GOTO 300
10000 'REM SUBRUTINA PARA MOSTRAR EN PANTALLA LA
    MATRIZ B
10005 PRINT CLR$;
```

```
10010 FOR K=1 TO H
10020 FOR J=1 TO W
10030 IF B(J,K)=1 THEN PRINT"*"; ELSE PRINT" ";
10035 PRINT B(K,J);
10040 NEXT J
10050 PRINT
10060 NEXT K
10070 RETURN
20000 'REM SUBRUTINA PARA IMPRIMIR LA MATRIZ A
20005 PRINT CLR$;
20010 FOR K=1 TO H
20020 FOR J=l TO W
20030 IF A(J,K)=1 THEN PRINT"*"; ELSE PRINT" ";
20035 'PRINT A(J,K);
20040 NEXT J
20050 PRINT
20060 NEXT K
20070 RETURN
30000 'REM ELABORAR LA DECISION
30010 IFN<2 OR N>3 THEN NXT=0:GOT0 30100 ;'REM MUERTE
    POR AISLAMIENTO O ASFIXIA
30020 IF C=0 AND N=3 THEN NXT=1 :GOTO 30100 'REM
    NACIMIENTO
30030 NXT=C'REM SIN CAMBIO
30100 RETURN
```

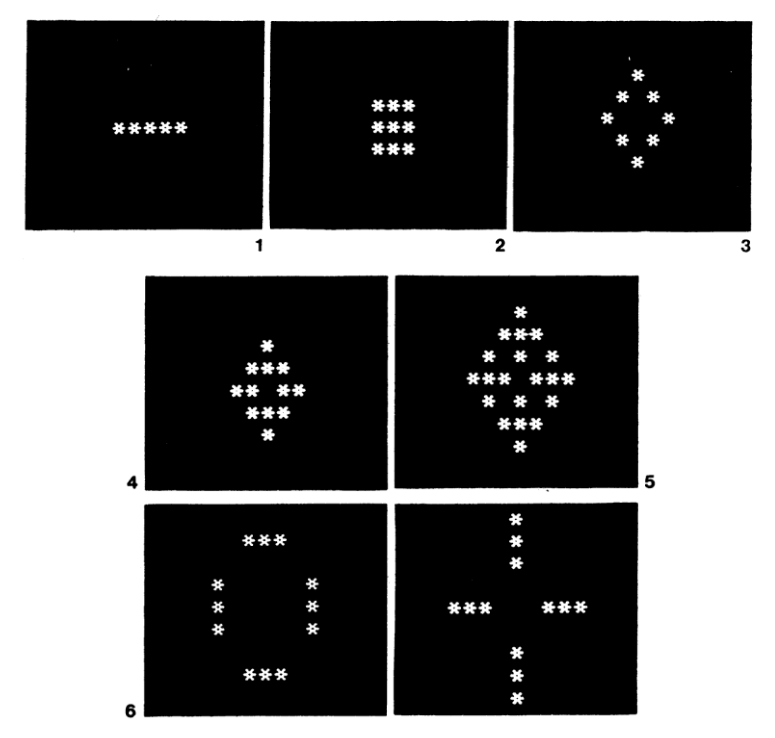
Examinemos una secuencia (abajo) más de cerca. Lo que ha pasado es:

1. Las dos células finales en 1 murieron, pero aparecieron tres más a cada lado, porque cada una de ellas tenía tres vecinos vivos.

2. Las esquinas sobrevivieron porque tenían tres vecinos, pero el resto se extingue por exceso de población.

3-6. En el anillo tuvieron demasiado éxito, y todas las células de dentro se extinguieron. Quedan cuatro conjuntos de *blinkers*. Un blinker es la representación de tres células en línea: las dos del final se extinguen por abandono, pero nacen dos más a cada lado; de este modo, la línea cambia

de vertical a horizontal y de nuevo a vertical y así sucesivamente. La configuración de cuatro blinkers se denomina “semáforos”.



Los entusiastas de “Life” han identificado cierto número de formas como ésta que se repiten cíclicamente. Una figura muy interesante es el “planeador”, una forma que se pasea moviéndose un cuadrado arriba y otro a lo largo de cada cuatro generaciones; para facilitar la visualización, lo hemos programado para que se desplazara horizontalmente.

x . .	. x .	x x .	. x .	. x
. x . .	x x . .	. . x .	. . x .	. . x
. x .	. x .	. x .	x x .	. . x
x x .	. x .	. x .	. x.	x x
0	1	2	3	4

Hay una forma bastante complicada llamada “cañón de planeadores”, que dispara un haz de planeadores cada treinta generaciones. Conway la ha utilizado para diseñar, lo que en principio parece un obstinado ejercicio intelectual, un computador que utiliza como materia prima, en lugar de impulsos eléctricos conducidos por cable, planeadores que se mueven en el interior de un computador. Un planeador representa un 1, la ausencia de un planeador representa un 0. Si chocan entre sí de forma apropiada, se eliminan ambos, de manera que puede construirse una puerta NOT. Disposiciones más complicadas permiten interaccionar dos haces de planeadores como si existieran puertas AND y OR. Otra configuración “comerá” planeadores, de modo que podremos deshacernos de los bits excedentes.

Con todas estas disposiciones se tiene la materia prima necesaria para un computador; aunque sea un computador construido con símbolos que operan dentro del hardware de un computador, pero que, al fin y al cabo, también constituyen un computador.

Tres en raya

“NANDC’ = Noughts AND Crosses (“Tres en raya”); fácil de jugar si se conoce. El robot de enseñanza que se describe en las páginas 152 y 153 puede ser considerado como una “rutina de visualización” para este tipo de programa. En lugar de dibujar O y X en la pantalla, el robot maneja piezas de damas sobre un tablero real.

```
10 REM ***************************
20 REM NOUGHTS AND CROSSES
30 REM ***************************
40 DIM T(8,3)
50 DIM U(9,4)
60 DIM C(9),B(9)
70 PRINT CHR$(12)
80 PRINT "*** NOUGHTS AND CROSSES ***"
90 PRINT
100 PRINT "USTED ES 'X', EL ORDENADOR ES 'O'"
110 PRINT "LO CREA O NO, USTED PUEDE GANAR"
120 X1=0:X2=0
```

```
130 REM INICIALIZA TABLAS 'P' y 'U'
140 FOR P=1 TO 8
150 FOR I=1 TO 3
160 READ T(P,I)
170 NEXT I
180 NEXT P
190 DATA 1,2,3
200 DATA 8,9,4
210 DATA 7,6,5
220 DATA 1,8,7
230 DATA 2,9,6
240 DATA 3,4,5
250 DATA 1,9,5
260 DATA 7,9,3
270 FOR S=1 TO 9
280 FOR J=1 TO 4
290 READ U(S,J)
300 NEXTJ
310 NEXT S
320 DATA 1,4,7,0
330 DATA 1,5,0,0
340 DATA 1,6,8,0
350 DATA 2,6,0,0
360 DATA 3,6,7,0
370 DATA 3,5,0,0
380 DATA 3,4,8,0
390 DATA 2,4,0,0
400 DATA 2,5,7,8
410 REM PARA UNA NUEVA PARTIDA
420 LET N=0
430 FOR S=1 TO 9
440 LET (C(S)=0
450 LET B(S)=0
460 NEXT S
470 REM MONEDA AL AIRE PARA VER QUIEN EMPIEZA
480 IF RND(l) < .5 THEN 510
490 PRINT "EMPIEZA USTED"
```

```
500 GOTO 350
510 PRINT "EMPEZA EL ORDENADOR"
520 GOTO 760
530 REM MUEVE EL JUGADOR
540 GOSUB 1230
550 INPUT "SU TURNO";M
560 LET F=I–1
570 IF M=INT(M) THEN 600
580 PRINT "JUGADA NO PERMITIDA-REPITA"
590 GOTO 530
600 IF M<I OR M>9 THEN 580
610 IF B(M) <> 0 THEN 580
620 REM ACTUALIZA LISTA C, ANALIZA SI HAY UN
    VENCEDOR
630 LET B(M)=F
640 FOR J=l TO 4
650 LET P=U(M,J)
660 IF P=0 THEN 700
670 LET C(P)=C(P)+F
680 IF C(P)=–3 THEN GOSUB 1230:GOTO 900
690 IF C(P)=3 THEN GOSUB 1230:GOTO 800
700 NEXT J
710 LET N=N+1
720 IF N=9 THEN GOSUB 1230:GOTO 930
730 IF F=1 THEN 530
740 REM JUEGA EL ORDENADOR
750 GOSUB 1230
760 GOSUB 950
770 PRINT "LA JUGADA DEL ORDENADOR ES:"
780 LET F=1
790 GOTO 620
800 REM PARTIDA ACABADA
810 PRINT "Y EL ORDENADOR HA GANADO"
820 X2=X2+1
830 PRINT
840 PRINT"RESULTADO. ORDENADOR";X2; ",USTED";X1
850 INPUT "PULSE 'SI' SI DESEA JUGAR DE NUEVO"; A$
```

```
860 IF A$<>"SI" THEN STOP
870 PRINT
880 PRINT "NUEVA PARTIDA"
890 GOTO 265
900 PRINT "USTED HA GANADO AL ORDENADOR"
910 X1=X1+1
920 GOTO 830
930 PRINT "PARTIDA DE PRUEBA"
940 GOTO 830
950 REM SELECCIONA JUGADA
960 FOR P=1 TO 8
970 IF C(P)=2 THEN 1030
980 NEXT P
990 FOR P=1 TO 8
1000 IF C(P)=–2 THEN 1030
1010 NEXT P
1020 GOTO 1070
1030 FOR I=1 TO 3
1040 LET M=T(P,I)
1050 IF B(M)=0 THEN 1220
1060 NEXT I
1070 FOR S=1 TO 9
1080 LET V(S)=0
1090 IF B(S)<>0 THEN 1150
1100 FOR J=1 TO 4
1110 LET P=U(S,J)
1120 IF P=0 THEN 1140
1130 LET V(S)=V(S)+1+ABS(C(P))
1140 NEXT J
1150 NEXT S
1160 LET V=0
1170 FOR S=l TO 9
1180 IF V(S)<=V THEN 1210
1190 LET V=V(S)
1200 LET M=S
1210 NEXT S
1220 RETURN
```

```
1230 REM IMPRIMA TABLERO
1240 PRINT
1250 PRINT" 1 2 3"; TAB(15);
1260 FOR A=1 TO 3
1270 GOSUB 1410
1280 NEXT A
1290 PRINT:PRINT
1300 PRINT " 8 9 4";TAB(15);
1310 A=8:GOSUB 1410
1320 A=9:GOSUB 1410
1330 A=4:GOSUB 1410
1340 PRINT:PRINT
1350 PRINT " 7 6 5";TAB(15);
1360 FOR A=7 TO 5 STEP –1
1360 GOSUB 1410
1380 NEXT A
1390 PRINT :PRINT
1400 RETURN
1410 IF B(A)=0 THEN PRINT". ";:RETURN
1420 IF B(A)=–1 THEN PRINT"X ";:RETURN
1430 PRINT"O ";:RETURN
```

Zombies

El juego de los “Zombies” es muy ingenuo. Se trata de esquivar estas horribles criaturas de modo que choquen contra los pilares, queden sin conocimiento y se autodestruyan; en caso contrario, se apoderan de usted y se lo comen.

```
10 REM ************'**'****'**
20 REM        ZOMBIES
30 REM ***************'******
40 REM 'GRAPH1' PONE EN MARCHA EL PLOTTING
50 REM 'GRAPHO' LO APAGA
60 REM 'PLOT X,Y,Z' COLOCA EL CARACTER CON CÓDIGO
   ASCII Z EN X,Y
70 CLEAR 1000
```

```
80 PRINT CHR$(12):GOSUB 640
90 GRAPH1
100 CLEAR
110 DIM B(12,22),Z(25,2),P(8),Q(8)
120 FOR N1=1 TO 8
130 READ P(N1),Q(N1)
140 NEXT N1
150 DATA 1,–1,1,0,1,1,0,1,–1,1,–1,0,–1,–1,0,–1
160 PLOT 74,57,53
170 PLOT 76,57,52
180 PLOT 78,57,51
190 PLOT 64,54,54
200 PLOT 76,54,88
210 PLOT 78,54,50
220 PLOT 74,51,55
230 PLOT 76,51,56
240 PLOT 78,51,49
250 FOR N1=1 TO 25
260 FOR N2=1 TO 2
270 Z(N1,N2)=0
280 NEXT N2
290 NEXT N1
300 Z1=0
310 FOR N1=1 TO 12
320 FOR N2=1 TO 22
330 B(N1,N2)=4
340 PLOT N2•2,N1•3,143
350 NEXT N2 360 NEXT NI
370 FOR N1=2 TO 11
380 FOR N2=2 TO 21
390 R=20*RND(10)
400 IF R>18.5 THEN GOTO 500
410 IF R<17.95 THEN GOTO 480
420 Z1=Z1+1
430 Z(Z1,1)=N1
440 Z(Z1,2)=N2
450 B(N1,N2)=2
```

```
460 PLOTN2•2,N1•3,ASC("Z")
470 GOTO 500
480 B(N1,N2)=1
490 PLOTN2•2,N1•3,ASC(" ")
500 NEXT N2
510 NEXT N1
520 X=5+INT(10•RND(15))
530 Y=3+INT(5•RND( 15))
540 B(Y,X)=3
550 PLOTX•2,Y•3,ASC("X")
560 FOR N1=Y–1 TO Y+l
570 FOR N2=X–1 TO X+1
580 IF ABS(Y–N1)+ABS(X–N2)=0 THEN GOTO 610
590 B(N1,N2)=1
600 PLOTN2•2,N 1•3,ASC(" ")
610 NEXT N2
620 NEXT N1
630 GOTO 770
640 PRINT"Usted acaba de aterrizar en ZOMBIE ISLAND"
650 PRINT
660 PRINT"Su única posibilidad de supervivencia es cazar"
670 PRINT"todos los ZOMBIES en trampas. Usted puede"
680 PRINT"indicar la dirección de su movimiento"
690 PRINT"de la siguiente manera:"
700 PRINT
710 PRINT"543"
720 PRINT"6X2"
730 PRINT"781":PRINT:PRINT
740 INPUT"Pulse RETURN para continuar" ;AA$
750 PRINT CHR$(12)
760 RETURN
770 REM
780 FOR N1=1 TOZ1
790 IF B(Z(N1,1),Z(N1,2)>=2 THEN GOTO860
800 FOR N2=N1 TO Z1
810 Z(N2,1)=Z(N2+1,1)
820 Z(N2,2)=Z(N2+1,2)
```

```
830 NEXT N2
840 Z1=Z1–1
850 GOTO 780
860 NEXT N1
870 PRINT
880 PRINT" Su turno";
890 INPUT A
900 IF A>8 THEN GOTO 920
910 IF A>=1 THEN GOTO 940
920 PRINT"Escoja un número entre 1 y 8"
930 GOTO 880
940 B(Y,X)=1
950 PLOTX•2,Y•3,ASC(" ")
960 Y=Y+Q(A)
970 X=X+P(A)
980 ON B(Y,X) GOTO 990,1020,990,1040
990 B(Y,X)=3
1000 PLOTX•2,Y•3,ASC("X")
1010 GOTO 1060
1020 PRINT"MMMMM SABROSO!—Come Come —
    PROTOPLASMA!"
1030 GOTO 1430
1040 PRINT"aaaaargh KASPLUTCH Directo al hoyo----";
1050 GOTO 1430
1060 Z2=l
1070 Z8=Z(Z2,1)
1080 Z9=Z(Z2,2)
1090 B(Z8,Z9)=1
1100 PLOT Z9•2,Z8•3,ASC(" ")
1110 Z8=Z8+SGN(Y–Z8)
1120 Z9=Z9+SGN(X–Z9)
1130 ON B(Z8,Z9) GOTO 1320,1280,1210,1140
1140 PRINT' 'KERSPLOSH---ZOMBIE va"
1150 FOR Z3=Z2 TO Z1
1160 Z(Z3,1)=Z(Z3+1,1)
1170 Z(Z3,2)=Z(Z3+1,2)
1180 NEXT Z3
```

```
1190 Z1=Z1–1
1200 GOTO 1370
1210 PLOT X•2,Y•3,42
1220 FOR MN=0 TO 150
1230 NEXT MN
1240 PLOT X•2,Y•3,32
1250 PLOT X•2,Y•3,ASC("Z")
1260 PRINT"Los ZOMBIES almuerzan a fin de cuenta!"
1270 GOTO 1430
1280 PRINT"EEK!--Ahí vienen los ZOMBIES!!";
1290 B(Z(Z2,1),Z(Z2,2))=2
1300 PLOTZ(Z2,2)•2,Z(Z2,1)•3,ASC("Z")
1310 GOTO 1360
1320 B(Z8,Z9)=2
1330 PLOT Z9•2,Z8•3,ASC("Z")
1340 Z(Z2,1)=Z8
1350 Z(Z2,2)=Z9
1360 Z2=Z2+1
1370 IF Z2<=Z1 THEN GOTO 1070
1380 IF Z1>=1 THEN GOTO 870
1390 PRINT
1400 PRINT"Enhorabuena! Usted ha escapado
1410 PRINT"los ZOMBIES han sido exterminados."
1420 GOSUB 1550
1430 PRINT" Otra partida ";
1440 REM
1450 INPUT A$
1460 IF A$="S" OR A$="SI" THEN PRINT CHR$(12): GOTO 90
1470 IF A$="N" OR A$="NO" THEN 1520
1480 PRINT"Conteste SI o NO":GOTO 1430
1490 GRAPH0
1500 RUN
1510 GOTO 1430
1520 GRAPH0
1530 PRINT CHR$(12)
1540 END
1550 FOR J=X+1 TO 79
```

```
1560 FOR HJ=0 TO 1
1570 PLOT J•2,Y•3,ASC("X")
1580 PLOT (J–1)•2,Y•3,ASC(" ")
1590 IF J•2>=78 THEN 1620
1600 NEXT HJ
1610 NEXT J
1620 PLOT J•2,Y•3,ASC(" ")
1630 RETURN
```

Apéndice 3

TABLA DEL CÓDIGO ASCII

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	↑@	↑A	↑B	↑C	↑D	↑E8	↑F	↑G	↑H	↑I	↑J	↑K	↑L	↑M	↑N	↑O»
0	NUL	SOH	STX	ETX	EOT	ENG	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
001	↑P	↑Q	↑R	↑S	↑T	↑U	↑V	↑W	↑X	↑Y	↑Z	↑[	↑\	↑]	↑↑	↑←
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	VS
010	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
2	SP	!	“	#/£	$	%	&	‘	(	)	*	+	,	_	.	/
011	48	49	50	51	52	53	54	55	56	57	58	69	60	61	62	63
3	Ø	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	80	81	82	83	84	85	66	87	88	89	90	91	92	93	94	95
5	P	Q	R	s	T	U	V	W	X	Y	Z	[	\	]	↑	←
110	96	97	96	99	100	>01	102	103	104	105	106	107	109	109	110	111
6		a	b	c	d	e	f	g	h	i	j	k	1	m	n	o
111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

PROCEDENCIA DE LAS ILUSTRACIONES

Los autores y editores quieren agradecer al editor de *Practical Computing* su autorización para reproducir los listados de programas del Apéndice 2.

Apple Computer Ltd.: Fig. 13
Art Directors Photo Library: Lámina 6
Chris Bidmead: Fig. 12
Paul Brierley: Lámina 2 (abajo)
British Telecom: Lámina 10
Calcomp Sanders: Lámina 4
Commodore: Lámina 3
Bob Chapman: Figs. 1, 2, 4, 5, 8, 11, 14, 16, 17, 18, 20, 21, 26, 27 y 29
Cray Research: Lámina 14
Departamento de Comercio e Industria: Lámina 1
Jeremy Enness: Fig. 24
Grundy & Northedge: Figs. 6, 7, 9, 25 y 28
General Electric Company PLC: Fig. 3
IBM: Lámina 13
Micro Control Systems: Lámina 8
Moving Picture Company: Lámina 7
Philips: Lámina 15
Ingram Pinn: Fig. 19
Science Museum, Londres: Fig. 30
Science Photo Library: Láminas 5 y 11
Smiths Industries: Lámina 9

También queremos expresar nuestro agradecimiento al profesor Alexander y al doctor Stonham, a la Universidad Brunel, a la Escuela Primaria Oíd Oak y a Mitsubishi Electric (UK) Limited por la generosa ayuda prestada en la realización de las fotografías.

INFORMATICA PARA TODOS
PETER LAURIE

Hasta hace poco tiempo el mundo de la informática estaba al alcance de muy pocos. La gente veía el computador como un instrumento de trabajo muy sofisticado y sólo manejable por verdaderos genios de la matemática.
Hoy día un computador, esa prolongación artificial del cerebro que es una herramienta para economizar trabajo y a veces, el más preciado juguete, ya es una "máquina" accesible, tanto por su precio como por su manejo.
Informática para todos es un libro técnico que está escrito en lenguaje corriente y que explica las virtudes o defectos del computador. A través de la lectura el autor "acompaña" al lector desde la elección y manejo de un computador personal hasta los más apasionantes descubrimientos en los campos del diseño asistido por computador (CAD), de la robótica y de la inteligencia artificial.
La lectura de este libro es de gran ayuda para todo el que desee adquirir un computador y utilizarlo en sus ratos de ocio o en su profesión.
Peter Laurie nació en Inglaterra en 1937. Estudió matemáticas y derecho en Cambridge. En 1976 se introdujo en el mundo de la electrónica y escribió un libro que causó gran impacto *(The Micro Revolution)*. En 1979 se convirtió en director de la revista *Practical computing*. Fue uno de los tres autores del best seller *The Computer Book* de la BBC. En 1980 fundó *Southdata*, empresa dedicada a la publicación de *software* y especializada en gestión de base de datos.

